

---

---

# **The 24<sup>th</sup> Winona Computer Science Undergraduate Research Symposium**

April 23, 2024  
2:00pm to 3:30pm  
Watkins 209

Winona State University  
Winona, MN

---

---

Sponsored by the Department of Computer Science  
at Winona State University



## Table of Contents

<b>Title</b>	<b>Author</b>	<b>Page</b>
<i>Investigating How Illumination Affects Infrared Gesture Recognition</i>	<b>Lindsey Arndt</b>	1
<i>Enhancing R2L Intrusion Detection Using Decision Trees</i>	<b>Stephen Sommer</b>	6
<i>A Data-Driven Approach to Keyboard Optimization</i>	<b>David Sommerfield</b>	11
<i>Elephant: LLM System for Accurate Recantations</i>	<b>Praise Chinedu-Eneh</b>	18

# Investigating How Illumination Affects Infrared Gesture Recognition

Lindsey Arndt

Computer Science Department

Winona State University

Winona, MN 55987

507.457.5000,

Lindsey.arndt@go.winona.edu

## ABSTRACT

The research aims to investigate the impact of illumination on the accuracy and efficiency of infrared gesture recognition systems. Specifically, the study seeks to explore whether infrared gesture sensors exhibit greater accuracy in low illumination environments compared to environments with high direct illumination.

The experimental setup includes a controlled environment with a moveable light source and a table holding the infrared gesture sensor. Fifty-six tests were conducted, with participants performing gesture recognition tasks under varying illumination levels ranging from 600 lux to 1200 lux. After experimentation, a t-test determined if there is a statistically significant difference in the accuracy of gesture recognition between the two groups.

The findings of this study show the threshold illumination is at 1070 lux. An increased illumination after the threshold causes gesture detection failure. Results compared to natural illumination from the sun, it is predicted there will be inconsistent gesture detection specifically under a clear sky in the afternoon.

## 1. INTRODUCTION

In recent years, the field of human-computer interaction (HCI) has witnessed a surge in the development and utilization of gesture recognition technologies for various applications ranging from gaming and virtual reality to smart home control and industrial automation. Among these technologies, infrared gesture sensors such as hands-free faucets, dryers, alarm clocks, and lamps have emerged as promising solutions due to their ability to accurately capture and interpret hand movements in real-time. However, the effectiveness of infrared gesture sensors is known to be influenced by environmental factors such as lighting conditions, which can significantly impact their performance and reliability in the real world.

The motivation for this study stems from the critical need to understand the environmental factors that affect the functionality of

infrared sensors, important to the future development of the technology.

To achieve this goal, the research focuses on analyzing the unified components that comprise infrared gesture sensors and their interactions with varying lighting conditions. By collecting comprehensive information about the design, operation, and sensitivity of infrared gesture sensors, the study aims to explain the mechanisms underlying their performance in different lighting environments.

The research conducted aligns with a study presented in the article titled "Gesture Recognition using Reflected Visible and Infrared Light Wave Signals" [6]. Employing experimental techniques aimed at manipulating environmental conditions, the gesture sensor underwent testing under conditions of high illumination juxtaposed with indirect low illumination. Interestingly, both this article and "A novel hand gesture recognition method based on infrared information" [7] reported no significant change in accuracy under low illumination conditions. In response to this finding, the gesture sensor underwent further testing under even brighter illumination.

The paper titled "Neuromorphic silicon retina array detects the direction of a moving object" [2] provides insights into the processing of infrared information within an infrared sensor. The study examines the detection of each finger using the infrared sensor. Given the project's focus on light source accuracy, a specific gesture, namely the upwards gesture, has been selected for testing with the sensor. The positioning of the light source behind the hand in front of the infrared sensor and individual testing on each side have been chosen for optimal evaluation accuracy.

## 1.2 Infrared Sensors

Infrared sensors emit light within the infrared spectrum. According to a newsletter from the International Commission on Non-Ionizing Radiation Protection, infrared, also known as thermal radiation, is above red visible light from 780 nanometers to 1000 micrometers on the electromagnetic spectrum [3]. Infrared naturally comes from solar radiation and fire. Artificial sources include lamps, heating devices, remote control, and the newly popular infrared saunas. The light can be seen as a dim red glow coming out of the sensor. Infrared sensors work by emitting infrared light and collecting reflections of light off objects that pass through. The reflection of light is collected by a light sensor. A program interprets any changes in the reflection as gestures. A specific gesture is determined by an algorithm that analyses the data. A popular algorithm for gesture recognition is Dynamic Time Warping. As mentioned in the Application of dynamic time warping algorithm for pattern similarity of gait, "The dynamic time warping (DTW)

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*Proceedings of the 24<sup>th</sup> Winona Computer Science Undergraduate Research Seminar, April 23, 2024, Winona, MN, US.*

algorithm was developed to solve the difficulties in analyzing pattern similarity for time-series data.” “DTW is known as the best algorithm for evaluating the similarity between two time-series data that vary in time frames as well as nonlinear time-series data.” [5] For gesture sensor recognition, the algorithm DTW can record the changes to the reflected infrared light and determine the direction of motion by comparing the pattern in the time-series data.

## 2. HYPOTHESIS

Infrared gesture sensors have greater accuracy in low illumination than high direct illumination.

## 3. METHODS

For this study, the researcher utilizes a programmable robot known as ClicBot, developed by KEYi Technology, and founded by Jianbo. The ClicBot is equipped with an infrared gesture sensor that offers users the ability to program and customize gesture recognition functionalities according to their preferences. To investigate the impact of environmental lighting conditions on the accuracy of infrared gesture recognition, the ClicBot is subjected to alternating environments. These environments include low illumination between 600 and 800 lux and high direct illumination between 900 and 1200 lux. As shown in Figure 1 and Figure 2, the illumination is placed directly in front of the gesture sensor. The light source moves forwards and backward to adjust to the illumination being tested as demonstrated by the x. The Illumination is calculated using a brightness meter placed directly in front of the infrared sensor before each test. The gesture is conducted 5 inches away from the sensor each time. Upon testing, 7 inches was the farthest away the sensor would detect motion. It was deemed reasonable to choose 5 inches as the distance because it provided sufficient proximity to the sensor for accurate gesture detection. Additionally, the arms on the assembled ClicBot extended 4 inches from the sensor, making it an ideal reference for gesture movement.

**Table 1. Summary of equipment used**

Equipment	Manufacturer	Quantity
ClicBot Brain	KEYi Technology	1
Illumination Kit (2 x 16" x 16") 50W / 5500K / 90 CRI LED Bulbs	RALENO	1
Table		1
Digital Illuminance Light Meter 0-200,000 Measurement Range Lux Meter	Dr. Meter	1

After conducting the indoor controlled experiment, the ClicBot was used to test gesture recognition outside in the natural environment. During the indoor experiment, the illumination source was focused and concentrated by using a single light in a dark room. In a natural setting, the illumination source comes from the sun. Sun rays are scattered filling a larger area causing illumination to hit front, back, and sides of the sensor. Establishing a direct pathway from the sun to the ClicBot posed a challenge, and although it wasn't flawless,

every effort was made to achieve the best possible outcome. The ClicBot was brought outside in an afternoon partly cloudy sky. A measurement of 1466 lux was obtained when the infrared sensor was facing the sun straight on, while 1090 lux was read when facing horizontal to the ground. A cloud covered for a brief period measuring at 260 lux. Each set of illuminations were tested 10 times with the same upward gesture as tested indoors.

Analysis of the data presented in Table 2 reveals significant variation in illumination levels and accuracy. As mentioned previously, there are numerous challenges in the natural environment to conduct a consistent controlled experiment. The scattering of sun rays through the atmosphere can change the illumination level even during an experiment. The readings were taken with the best effort. On a cloudy day, there may not be any reason for concern with gesture recognition. However, there is a chance recognition accuracy will be inefficient. It is reasonable to conclude that numerous factors contribute to these fluctuations in lux readings. Consequently, caution should be exercised when interpreting results obtained from natural settings.

**Table 2. Natural Illumination Results**

Illumination Lux	% Accurate
1466	50%
1090	40%
260	100%



**Figure 1. Experiment room**

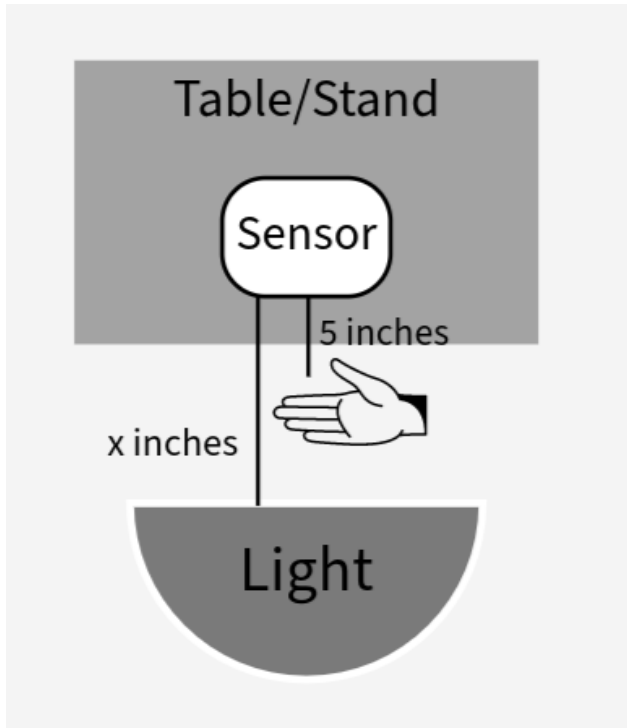


Figure 2. Layout of experiment

### 3.1 Analysis

A logistic regression was performed to ascertain the effects of illumination on the likelihood that greater accuracy in low illumination than high direct illumination. A logistical regression is performed because there are two different value means in which the data is normally distributed, and variances are unknown. A p-value is significant when the value is below .05%. A software called JMP was used to conduct the method by comparing the illumination in lux to the response classification. The logistic regression model was statistically significant ( $1, N = 55 = 44.55, p < .0001$ ). The model explained 15.0% of the variance in gesture accuracy and correctly classified 80.0% of cases. The logistical regression results are shown in Table 3. As shown in Table 3, the standard error is 0.0409851. This standard error means that, on average, the estimated coefficient for the independent variable illumination level in the logistical regression model can be expected to vary by approximately 0.0409851 units from the true population coefficient. If the study were to be replicated, the estimated coefficient for illumination level threshold to fall within  $\pm 0.0409851$  lux of its current value about 95% of the time. The results went into the creation of Figure 2 showing the probability of gesture recognition in correlation with illumination levels.

Table 3. Parameter estimates

Term	Estimate	Std Error	P-Value < .0001
<b>Illumination</b>	-0.09404	0.0409851	0.0215

Figure 3 is a graph representing the probability of accurate gesture recognition on illumination level using the logistic plot created by the JMP software. The x-axis holds the illumination level in lux and the y-axis holds the probability of accurate gesture recognition. As shown, there is a drop in probability of accurate gesture recognition from 100% accuracy at 1000 lux to 0% accuracy at 1200 lux. The blue line in figure 3 shows this threshold where gesture recognition fails to recognize any gesture. Keeping that in mind, the graph shows the trend. Tracking the threshold line, looking at 600 lux, it is presumable that gesture recognition is 100% as for all the lux range up until 1000 lux.

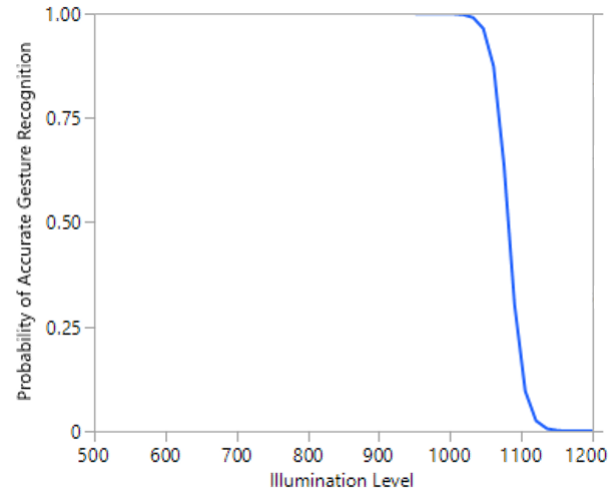


Figure 3. Probability of accurate gesture recognition on illumination level

To maintain consistency and accurately track gesture recognition accuracy, the ClicBot and light source stays at the same height. The light is facing the ClicBot straight onto the sensor. The light is moved to change its distance to the ClicBot for the purpose of adjusting the illumination. Upon recognizing a gesture, the ClicBot emitted a sound corresponding to the recognized gesture using a simple if-then statement. The response was categorized as accurate if the gesture is correctly identified, inaccurate if there is a misinterpretation, or fail to process if the gesture is not recognized.

Data collection involved recording the accuracy of gesture recognition for each gesture in both low illumination and high direct illumination environments. This data is documented in a spreadsheet, as illustrated in Table 2. Subsequently, a logistical regression was conducted to visualize the accuracy of gesture recognition in each environment.

### 3.2 Data Collection

Data was collected shown in the data collection sheet in Table 4. The distance is measured in inches from the illumination source to the infrared sensor. This distance is correlated with the brightness of illumination. The closer the light is, the higher the illumination is. We use distance the gauge the illumination because the light itself is unable to adjust its brightness. The illumination in lux is taken from the Digital Illuminance Light Meter. The measurement

is taken at the very front of the infrared sensor at the same location each time. The illumination written down is the number that showed most prominently on the scanner. The illumination would adjust by 10 to 15 lux, so a medium was used for monitoring purposes. Accurate Response percentage shows the percentage of accurate gesture recognition for each 5 illumination tests. This response percentage was taken by recording how many gestures were accurately recognized or failed to be recognized. Illumination level categorized the level most fitting in correlation to the illumination measurement in lux. Low illumination categorized lux from 530 to 790 and anything below this lux until 100 lux not tested in this study. This illumination could be seen outside naturally on a cloudy or partly sunny day. High illumination categorized the lux between 970 and 1174 also including any lux above this. High illumination can be seen in nature on a partly cloudy to full sun day, specifically in the afternoon.

**Table 4. Data collection sheet**

Distance (inches)	Illumination (lux)	Accurate Response %	Illumination Level
12.25	530	100	low
11.12	636	100	low
9.5	720	100	low
8.75	790	100	low
7.12	970	100	high
7.865	1070	100	high
7.225	1075	40	high
7.125	1080	40	high
7.25	1096	40	high
6.65	1111	0	high
6.55	1174	0	high

## 4. CONCLUSION

Based on the statistically significant results from the logistical regression, it can confidently be said that gesture recognition is more accurate in low illumination than high illumination. This statement is concurrently validated for outdoor natural lighting. As a result, there should be caution when utilizing infrared sensors outdoors.

As concluded, there is a challenge with sensing gestures in high illumination indoor and outdoor. Compared to turning up the brightness on a photo, excessive illumination can cause information to be washed out. This abundance of light can overwhelm the sensor with information, representing the reflections from hand movements insignificant. The proposed issue also relates to why there is no incorrect gesture detected; rather, there is simply a failure to detect any gesture. One potential solution to this problem could involve the same principle as enhancing photos. Increasing contrast causes lines to become more prominent. However, implementing this solution is easier said than done. If it proves to be impractical, it may be advisable to suggest regulations on the

usage of these devices. Regulations can specify when and where infrared gesture sensors should be employed. For example, there should be a warning on devices for minimizing exposure to direct sunlight while using the product.

## 5. ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed. Thank you to my advisors on this project Mingrui Zhang and Sudharsan Iyengar for their help and guidance. Another big thank you to Tisha Hooks for assisting with data analysis on this project and to Collin Engstrom for naming the ClicBot for experimentation Marcy.

## 6. REFERENCES

- [1] Aupal, M. (2015-2016). What is an IR Sensor. TeachEngineering. Retrieved from [https://www.teachengineering.org/lessons/view/mis\\_sensor\\_lesson01](https://www.teachengineering.org/lessons/view/mis_sensor_lesson01)
- [2] Cha, Y., Lee, J., Kim, Y., Jeon, Y., Lee, D. S., & Lee, H. J. (2021). Neuromorphic silicon retina array detects the direction of a moving object with sub-millisecond latency. *Proceedings of the National Academy of*
- [3] International Commission on Non-Ionizing Radiation Protection (CNIRP). (n.d.). Infrared Radiation. Retrieved from <https://www.icnirp.org/en/frequencies/infrared/index.html#:~:text=Common%20natural%20sources%20are%20solar,infrared%20saunas%20for%20health%20purposes.>
- [4] Keyi Technology. (n.d.). Module. Retrieved from <https://keyirobot.com/pages/module>
- [5] Lee HS. Application of dynamic time warping algorithm for pattern similarity of gait. *J Exerc Rehabil.* 2019 Aug 28;15(4):526-530. doi: 10.12965/jer.1938384.192. PMID: 31523672; PMCID: PMC6732547.
- [6] Yu, L., Abuellaz, H., Islam, M. Z., O'Hara, J. F., Crick, C., & Ekin, S. (2020). Gesture Recognition using Reflected Visible and Infrared Light Wave Signals. *ResearchGate*. Retrieved from [https://www.researchgate.net/publication/343005783\\_Gesture\\_Recognition\\_using\\_Reflected\\_Visible\\_and\\_Infrared\\_Light\\_Wave\\_Signals](https://www.researchgate.net/publication/343005783_Gesture_Recognition_using_Reflected_Visible_and_Infrared_Light_Wave_Signals)
- [7] Sciences. Retrieved from <https://www.pnas.org/doi/full/10.1073/pnas.2021077118>
- [8] Wang, J., Zhang, J., Zheng, Z., Xie, L., Zhang, C., & Li, Y. (2022). A novel hand gesture recognition method based on infrared information. *Optik*, 254, 166672. DOI: 10.1016/j.ijleo.2021.166672

# Enhancing R2L Intrusion Detection Using Decision Trees

Stephen Sommer

Computer Science Department  
Winona State University  
steven.sommer@go.winona.edu

## ABSTRACT

This research investigates how Decision Trees, a type of machine learning tool, can improve Intrusion Detection Systems (IDS) to identify Remote to Local (R2L) attacks. Traditional Intrusion Detection Systems (IDS) are struggling to cope with the growing complexity and frequency of attacks. This research explores whether Decision Trees and using it with combined datasets can provide a better solution. We utilized two widely available datasets, KDD Cup 1999, and NSL-KDD, to assess the effectiveness of Decision Trees. By applying Decision Trees to each dataset individually and then to a combined version, the combined approach resulted in an IDS with 99.93% accuracy, a 91% detection rate for R2L attacks, and only an 8% false alarm rate. While the improvement in overall accuracy over the individual datasets (99.90% for KDD Cup 1999 and 99.83% for NSL-KDD) is modest, it lays the groundwork for further refinements and potentially even greater accuracy gains in the future.

## Categories and Subject Descriptors

D.C.2.0 [Computer-Communication Networks]: General – security and protection

K.6.5 [Management of Computing and Information Systems]: Security and Protection – intrusion detection.

## General Terms

Algorithms, Performance, Experimentation, Security, Verification.

## Keywords

Remote to Local (R2L), Intrusion Detection Systems (IDS), Artificial Intelligence (AI), Machine Learning, Decision Tree Dataset Analysis, Cloud Computing Security, KDD Cup 1999, NSL-KDD, Intrusion Detection Accuracy.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*Proceedings of the 24<sup>th</sup> Winona Computer Science Undergraduate Research Seminar, April 23, 2024, Winona, MN, US.*

## 1. INTRODUCTION

In the age of advancing technological landscape, characterized by the rise of artificial intelligence and big data, the internet is playing a vital role in nearly every aspect of our lives, from conducting global business deals, to attending virtual classrooms, and stay in touch with family and friends. However, this increasing reliance on digital networks also exposes us to cybersecurity threats and attacks. Cyber-attacks fall into four main categories such as Denial-of-Service (DoS), Remote-to-Local (R2L), Unauthorized Access to Local Superuser (Root) Privileges (U2R) and Probing. To protect us from these attacks there are Intrusion Detection Systems. An Intrusion Detection System is a security tool designed to monitor network or system activities for malicious or suspicious behavior. It works by analyzing incoming traffic, identifying patterns that may indicate unauthorized access or potential threats, and alerting system administrators or security personnel to take appropriate action.

Traditionally, Intrusion Detection Systems (IDS) relied on predefined signatures to identify known attack methods, but, with the constantly evolving tactics of cybercriminals, this approach often proves inadequate. However, through the integration of Artificial Intelligence (AI) technologies, IDS can now harness advanced algorithms and machine learning techniques. This enables them to enhance their detection capabilities significantly, empowering them to adapt and respond more effectively to the ever-changing landscape of cyber threats. Despite these advancements, AI-enhanced IDS face specific challenges. The training of these systems need a vast amount of labeled data, which can be challenging and time-consuming to gather. Moreover, ensuring the transparency and explainability of AI decisions is vital for preserving trust and security. Biases in the training data can result in inaccurate detections or even discrimination.

This research focuses on using Decision Tree Classifiers to improve how Intrusion Detection Systems detect Remote-to-Local (R2L) attacks. In an R2L attack, an attacker sends data packets over a network to gain unauthorized access to a machine as a regular user. Unlike attacks that target the network itself, R2L attacks try to exploit weaknesses in a system's security to gain access. Attackers might use brute-force password cracking, exploit software vulnerabilities, or trick users into running malicious code. Their goal is to steal information, damage data, or prevent authorized users from accessing the system.

## 2. HYPOTHESIS

By incorporating multiple datasets into a decision tree classifier, intrusion detection systems can achieve greater accuracy, leading to a higher rate of R2L attack detection with fewer false alarms.

### 3. METHODOLOGY

#### 3.1 Datasets

This research evaluates an Intrusion Detection System (IDS) using two publicly available datasets: KDD Cup 1999 and NSL-KDD. Both datasets contain labeled network traffic data, including normal traffic, various attack types and the features to look out for to indicate the possibility of an attack.

- **KDD Cup 1999:** The KDD Cup 1999 dataset serves as a standard benchmark dataset extensively used for evaluating intrusion detection systems. It contains a substantial volume of network traffic data, encompassing both normal and various types of attacks. The dataset contains a total of 494,021 data points, with 97,278 labeled as anomalies and 396,743 as normal.
- **NSL-KDD:** The NSL-KDD dataset is another widely adopted dataset in the realm of intrusion detection. It represents an enhanced version of the original KDD Cup 1999 dataset, addressing certain limitations such as redundancy and imbalance in class distribution. The NSL-KDD dataset contains a total of 2,406,084 data points. Of these data points, 2,396,857 are labeled as normal and 9,227 are labeled as anomalies.

#### 3.2 Combined Dataset

To enhance the diversity and resilience of our training data, we combined the KDD Cup 1999 and NSL-KDD datasets into a unified dataset. This merged compilation broadens the scope of network traffic samples, encompassing a variety of attack types and normal behaviors. Sticking to a single dataset for training could limit a model's adaptability to unforeseen data. By fusing the KDD Cup 1999 and NSL-KDD datasets, our consolidated dataset provides a more inclusive depiction of network traffic behaviors. This, in turn, facilitates the development of machine learning models capable of discerning a wider array of anomalies and intrusions. The diversity within our training data fosters the creation of models that can adeptly adjust to evolving cyber threats and fluctuations in network landscapes. Moreover, the combined dataset enables researchers and practitioners to explore the similarities and differences between the two datasets, potentially uncovering valuable insights into the characteristics of network attacks and normal traffic across different contexts and time periods.

#### 3.3 Preprocessing the Data

The initial phase of this research entails parsing the `kddcup.names` file to extract and categorize its features for analysis. These features encompass various indicators relevant to security assessments, such as duration, source bytes, destination bytes, number of failed login attempts, number of compromised or hacked systems in a network, a user has obtained root access or escalated privileges, switch user attempted, and logged in as a guest, among others. These attributes provide insights into potential R2L (Remote-to-Local) attack scenarios, highlighting patterns associated with prolonged connection times, data transmission volumes, brute force password attacks, compromised states, and unauthorized access attempts, including tactics like guest logins. Subsequently, the analysis extends to different types of R2L attacks, including 'ftp\_write', 'guess\_passwd', 'imap', 'multihop', 'phf', 'spy', 'warezclient', and 'warezmaster'. Each of these attack types represents distinct forms of security breaches or suspicious activities, such as unauthorized

file transfers via FTP, repeated password guessing attempts, illicit access to email servers, suspicious routing behaviors, exploitation of web server vulnerabilities, covert surveillance activities, and involvement in the distribution of pirated software, potentially involving malware dissemination. These insights aid in identifying and understanding the diverse tactics employed by attackers in R2L attack scenarios. Following initial steps, such as addressing missing values and multicollinearity, categorical features are converted into numerical representations via feature mapping. Irrelevant features are then removed to streamline the dataset. Subsequently, the dataset is split into subsets for normal records and R2L attack records from the other types of cyber-attacks, facilitating machine learning model training. This includes feature identification and target label assignment, followed by division into training and testing sets using `sklearn's train_test_split` function, ensuring accurate model evaluation.

#### 3.4 Decision Tree Classifier

The Decision Tree Classifier is a tool used in machine learning to create models that can classify data into different categories. It relies on a decision tree structure, which is like a flowchart with nodes representing decisions and branches representing possible outcomes. This classifier is particularly useful for tasks like intrusion detection in computer networks because it's easy to understand how it reaches its conclusions, and it can handle different types of data, like numbers and categories, effectively.

In this process, we use a specific implementation of decision trees called `DecisionTreeClassifier` from a Python library called `scikit-learn`, often abbreviated as `sklearn`. This library provides a straightforward yet powerful way to build models using decision trees. We start by training the model, which means we show it examples of network traffic labeled as normal or malicious, and it learns from these examples. Then, we evaluate how well it performs on data it hasn't seen before, using a separate testing dataset.

To get started, we import several important libraries we'll need, such as `DecisionTreeClassifier` from `sklearn`, along with others like `cross_val_score` for cross-validation, `train_test_split` to split our data, `accuracy_score` to measure accuracy, and `pandas` for data manipulation. We also do some preprocessing on our data, like converting categorical features into a format the model can understand using one-hot encoding, and splitting our data into training and testing sets, with 70% of the data for training and 30% for testing.

Next, we create an instance of the `DecisionTreeClassifier` class. Then, we use a technique called cross-validation on our training data. This involves splitting our training data into 10 smaller subsets, training the model on 9 of them, and testing it on the remaining one. We do this process 10 times, each time with a different subset held out for testing. This gives us a more robust estimate of how well our model will perform on unseen data.

After cross-validation, we train the model on the entire training set and then use it to make predictions on the test set. Finally, we can evaluate the performance of our model by comparing its predictions to the actual labels in the test set. This gives us a measure of how accurate our model is at classifying network traffic as either normal or malicious.



## 4. RESULT AND ANALYSIS

### 4.1 Results

The results indicate that the use of a combined dataset significantly enhances intrusion detection accuracy. The IDS achieves high accuracy, detection rate, and maintains a low false alarm rate. As shown in Table 1:

**Table 1. Comparison of Intrusion Detection Performance Across Datasets**

Name	Accuracy	Detection Rate	False Alarm Rate
KDD Cup 1999	99.90%	83%	8%
NSL-KDD	99.83%	89%	10%
Combined Datasets	99.93%	90%	8%

These results demonstrate the superiority of the IDS utilizing the combined dataset over those relying on individual datasets alone. The combined dataset shows promise for enhanced intrusion detection. While the current improvement in accuracy over individual datasets (99.93% vs. 99.83-99.90%) is modest, it lays the groundwork for further refinements and potentially even greater accuracy gains in the future. Moreover, the detection rate for the combined dataset reaches 90% representing a notable improvement over the individual datasets. Additionally, the combined dataset achieved a noteworthy 90% detection rate, surpassing the performance of individual datasets. Importantly, the false alarm rate remained low at only 8%, demonstrating the IDS's ability to differentiate between normal and abnormal network activity. These findings strongly support the value of integrating diverse datasets to improve the overall performance of intrusion detection systems.

### 4.2 Classification Report

Analyzing the classification reports for KDD-CUP 1999, NSL-KDD, and the combined dataset reveals generally promising performance across all. Notably, most reports show consistently high accuracy (often 1.00) for normal traffic classification, indicating the model's ability to accurately identify normal network activities.

Precision tells us how accurate our "malicious" detections are, indicating the proportion of times the model flagged something and it actually turned out to be a true threat.

Recall, on the other hand, focuses on how well the model identifies all malicious activities, ensuring it doesn't miss any threats.

The F1-score provides a single measure of how well the model identifies true threats. It considers both precision (accurately identifying malicious traffic) and recall (catching most of the malicious activity). The high precision and recall scores we see suggest the model excels at both accurately flagging threats and minimizing missed threats.

Support shows the number of instances in each category (malicious, normal, etc.). This helps us interpret precision and recall, especially for less common activities.

Classification Report for KDD-Cup 1999				
	precision	recall	f1-score	support
ftp_write.	0.67	0.40	0.50	5
guess_passwd.	1.00	1.00	1.00	17
imap.	0.33	0.67	0.44	3
multihop.	0.40	0.67	0.50	3
normal.	1.00	1.00	1.00	29172
phf.	1.00	1.00	1.00	1
warezclient.	0.97	0.99	0.98	314
warezmaster.	1.00	0.71	0.83	7
accuracy			1.00	29522
macro avg	0.80	0.81	0.78	29522
weighted avg	1.00	1.00	1.00	29522

**Figure 2.1 Classification Report for KDD-CUP 1999**

Classification Report for NSL-KDD				
	precision	recall	f1-score	support
ftp_write	1.00	1.00	1.00	1
guess_passwd	1.00	0.89	0.94	9
imap	0.67	1.00	0.80	4
multihop	0.00	0.00	0.00	6
normal	1.00	1.00	1.00	20196
phf	1.00	1.00	1.00	1
spy	0.00	0.00	0.00	2
warezclient	0.96	0.96	0.96	275
warezmaster	0.62	0.62	0.62	8
accuracy			1.00	20502
macro avg	0.69	0.72	0.70	20502
weighted avg	1.00	1.00	1.00	20502

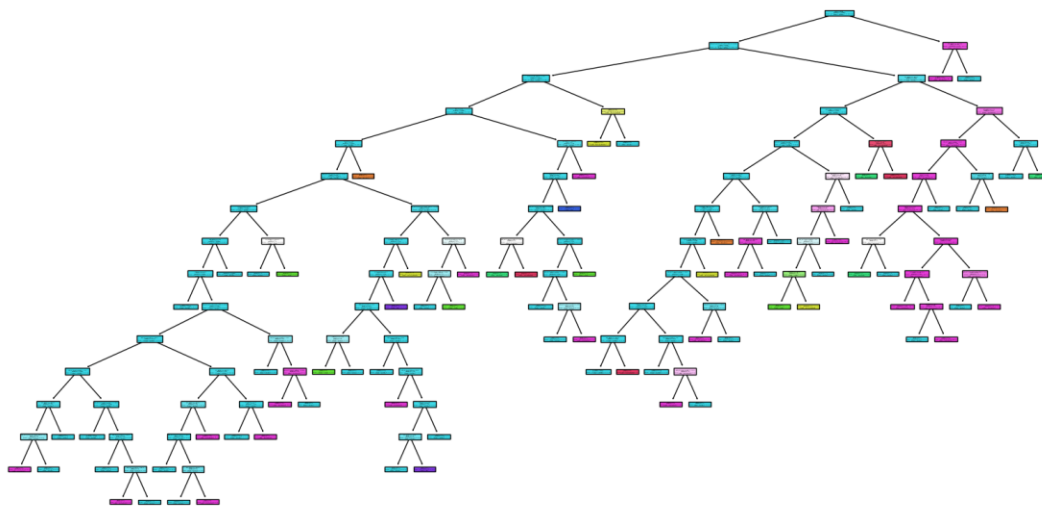
**Figure 1.2 Classification Report for NSL-KDD**

Classification Report for Combined Dataset				
	precision	recall	f1-score	support
ftp_write.	0.67	0.40	0.50	5
guess_passwd.	0.89	1.00	0.94	17
imap.	0.50	0.33	0.40	3
multihop.	0.50	0.67	0.57	3
normal.	1.00	1.00	1.00	29172
phf.	1.00	1.00	1.00	1
spy.	0.00	0.00	0.00	0
warezclient.	0.97	1.00	0.98	314
warezmaster.	1.00	0.71	0.83	7
accuracy			1.00	29522
macro avg	0.73	0.68	0.69	29522
weighted avg	1.00	1.00	1.00	29522

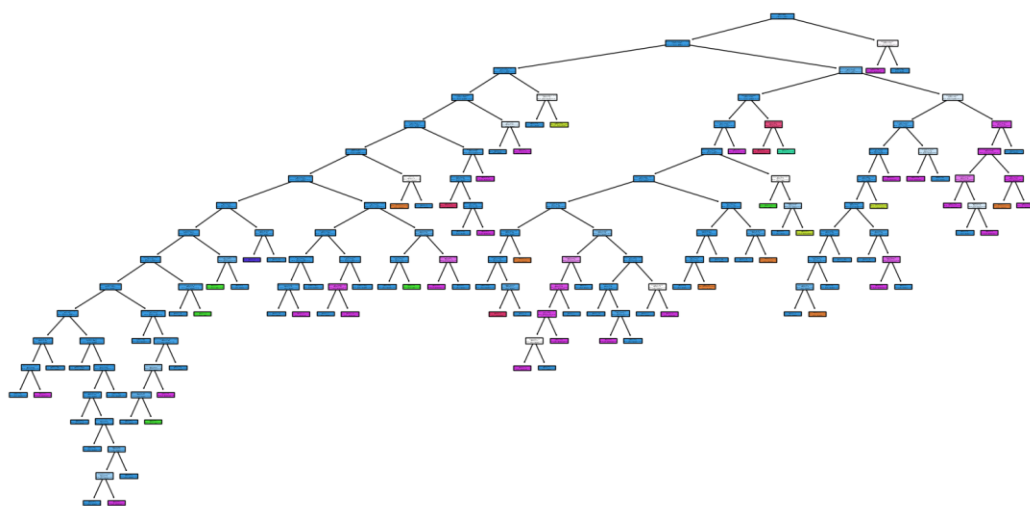
**Figure 1.3 Classification Report for the Combined Dataset**

### 4.3 Analysis

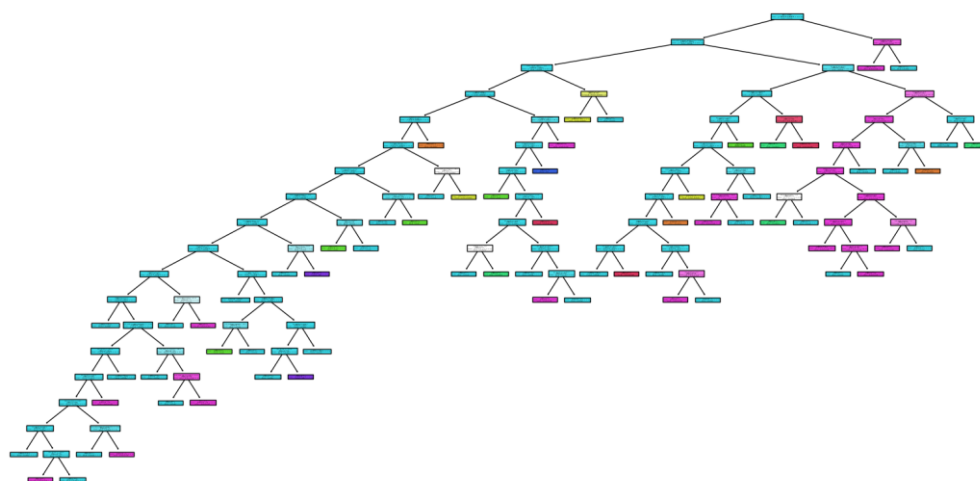
In this research, I initially explored various machine learning algorithms but ultimately focused on Decision Trees to delve into how Artificial Intelligence reaches decisions, particularly in Intrusion Detection Systems (IDS). As I constructed the decision tree and merged datasets, I observed that the KDD-Cup 1999 set had 76, NSL-KDD had 85 and the Combined Decision Tree had 66 leaves.



**Figure 2.1. Decision Tree from KDD Cup 1999**



**Figure 2.2 Decision Tree from NSL-KDD**



**Figure 2.3 Decision Tree – Combined Dataset**



## 5. CONCLUSION

In conclusion, this research has shed light on the potential of decision trees as a valuable tool for enhancing Remote to Local (R2L) intrusion detection. By leveraging the combined strengths of multiple datasets, like the KDD Cup 1999 and NSL-KDD datasets used in this research, the research demonstrates a slight enhancement in detection rates for R2L attacks.

This is particularly noteworthy when considering the low false positive rate achieved by the model. These findings pave the way for the development of more robust intrusion detection systems capable of proactively mitigating R2L attacks and bolstering overall cybersecurity posture. However, there are still challenges to overcome. One major hurdle is the evolving nature of cyber threats. Attackers continuously develop new techniques, and decision trees trained on historical data may struggle to identify novel attack patterns. Additionally, the effectiveness of decision trees can be hampered by data quality issues and class imbalance within intrusion detection datasets. Another challenge is the limited time available for research, which can restrict the scope and depth of investigation.

To address these challenges, future research can explore strategies for continuous learning and adaptation of the decision tree model. This could involve incorporating real-time threat intelligence feeds or employing online learning algorithms that can update the model as new data becomes available. Furthermore, investigating techniques to handle class imbalance, such as oversampling or cost-sensitive learning, could improve the model's ability to detect rare R2L attacks. By implementing these strategies, we can strive towards developing intrusion detection systems that remain vigilant in the face of ever-changing cyber threats.

## 6. ACKNOWLEDGMENTS

I would like to express my sincere gratitude to Dr. Mingrui Zhang and Dr. Sudharasan Iyengar for their guidance and support throughout this research. Their mentorship in both planning and executing this project has been invaluable. I am also grateful to Dr. Collin Engstrom for his insightful teachings on machine learning, which have greatly inspired my pursuit of knowledge in this field.

Special thanks are due to Eric Wright and Nadia Miranda for their support and assistance, not only in my research but also throughout my undergraduate studies. Lastly, I want to extend my heartfelt appreciation to my girlfriend, Micayla. Her support and encouragement throughout this research journey, despite the technical nature of the subject matter, has been a constant source of motivation.

## 7. REFERENCES

- [1] S.Mukkamala, A. H. Sung, and A. Abraham, "Intrusion detection using an ensemble of intelligent paradigms," *J. Netw. Comput. Appl.*, vol.28 no.2, pp. 167-182, Apr. 2005
- [2] KDD CUP 1999 Data. (n.d.). <https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
- [3] NSL-KDD Data. (n.d.). <https://www.kaggle.com/datasets/hassan06/nslkdd/data>
- [4] Jjj Tsai, C., Hsu, Y. F., Lin, C., & Lin, W. (2009). Intrusion detection by machine learning: A review. *Expert Systems With Applications*, 36(10), 11994–12000. <https://doi.org/10.1016/j.eswa.2009.05.029>
- [5] GeeksforGeeks. (2022, January 14). Intrusion detection system using machine learning algorithms. <https://www.geeksforgeeks.org/intrusion-detection-system-using-machine-learning-algorithms/>
- [6] Samunnisa, K., Kumar, G. S. V., & Madhavi, K. (2023). Intrusion detection system in distributed cloud computing: Hybrid clustering and classification methods. *Measurement: Sensors*, 25, 100612. <https://doi.org/10.1016/j.measen.2022.100612>.
- [7] ACM SIG PROCEEDINGS template. <http://www.acm.org/sigs/pubs/proceed/template.html>
- [8] Samunnisa, K., Kumar, G. S. V., & Madhavi, K. (2023). Intrusion detection system in distributed cloud computing: Hybrid clustering and classification methods. *Measurement: Sensors*, 25, 100612. <https://doi.org/10.1016/j.measen.2022.1>

# A Data-Driven Approach to Keyboard Optimization

David Sommerfield  
davidssommerfield.zg@gmail.com  
Winona State University  
Computer Science Department  
Winona, Minnesota, USA

## Abstract

Keyboard layout optimization has historically relied on observations and theory instead of statistically robust modeling. QWERTY, a product of these observations, was popularized by its usage on typewriters in the 1870s and remains the de facto keyboard layout today. Furthermore, the search space for layouts grows factorially, making it impossible to search exhaustively. To address these two issues, we propose a layout-creation method that uses metaheuristic optimization, corpus data from the web, and real-world typing data. The typing data and corpus are decomposed into sequences of two and three, referred to as bistrokes and tristrokes. From these sequences we construct a regression model for predicting typing speed by using average typing time, corpus frequency, and positional features. An objective function is established for the optimization by multiplying the sum of each tristroke’s occurrence count by its predicted typing time, effectively estimating the layout’s typing time across the corpus. Then, we apply customized simulated annealing to converge towards a near-optimal keyboard arrangement. The result is a keyboard layout with an estimated 6% improvement over QWERTY’s typing time.

## 1 Introduction

The most widely adopted keyboard layout, QWERTY, was first created for typewriters by Christopher Sholes in the early 1870s. It was further developed and popularized with the success of the Remington No. 2 typewriter, which featured the layout [13]. The design of the typewriter informed the design of computer keyboards, keeping the QWERTY layout. Today, 150 years since its invention, QWERTY is the most common keyboard layout and by extension the most common means of human-computer interaction, extending even to mobile devices. The continued usage of QWERTY has led to concerns about its antiquated design posing risks of injury or discomfort for its users [1]. Having recognized this issue, many have sought to create keyboard layouts that mitigate QWERTY’s shortcomings.

In the 1930s, Dvorak proposed more rigorous methods of evaluation for layout optimization, marking perhaps the first venture into using statistics for this problem. Numerous ergonomic factors have since been identified from studies

like these. However, the effect of these ergonomic factors is seldom numerically measured to determine an evaluation model. Instead, researchers tend to rely on these observations to inform theoretical models. There are several standard practices derived from these observations: optimizing the distance between keys, choosing columns according to perceived finger dexterity, limiting the occurrence of so-called single-finger bigrams, assuming perfect touch typing, and considering relative character frequencies [12].

Despite advancements since the proliferation of QWERTY, the field still faces challenges in quantifying the precise impact of each ergonomic factor on typing efficiency and user comfort. Underscoring these considerations is the challenge of the search space itself; optimizing for  $n$  keys creates a search space of  $n!$  potential layouts, making keyboard optimization a non-trivial problem in combinatorial optimization. Consequently, researchers have used various metaheuristic approaches to effectively explore this search space, including simulated annealing [10], ant colony optimization [5], swarm optimization [14], genetic algorithms [9], and more.

## 2 Methodology

### 2.1 Data Processing

Two datasets are used to estimate the typing time. The first dataset, the iWeb corpus [3] is one of the largest available collections of English text and uses a systematic selection of websites to ensure data quality. This corpus data is used for frequency analysis of characters and character sequences. The second data set, The 136M Keystrokes Dataset from Aalto University [4], contains typing test performance data totaling about 8,228 hours from 168,000 participants. Participant metadata enables a controlled analysis of four layouts: AZERTY, Dvorak, QWERTY, and QWERTZ.

Several steps are taken to ensure the keystroke data produces a quality analysis. First, data undergoes normalization, where it is segmented into distinct sessions and users and occasional errors in the source are corrected. Then, the data is updated to reflect the accuracy of each keystroke through approximate string matching.

When determining string correctness, three types of typing errors must be accounted for: insertion, deletion, and swapping [11]. For each typing session, a keystroke validity record is generated. Validity is updated on a per-window basis to account for navigation within the typing session using arrow keys or backspacing. Anytime the user makes a

**Table 1.** Types of Error Handled by Approximate String Matching

Typing Error	String typed
None	But thank you for the offer
Insertion	But thankk you for the offer
Deletion	But tha k you for the offer
Substitution	But thabk you for the offer

move with the arrow keys or backspaces, the typing record is updated to reflect the correctness of each keystroke typed in the window before it; the corrected window is appended to the keystroke validity record, then a new window is created starting from the last arrow key or backspace typed. Once the last keystroke has been processed, the remaining window is updated to reflect correctness and pushed to the keystroke validity record. The validity record is then used to determine the type of errors made so that the analysis can only include correctly typed keystrokes. Insertion and replacement errors need no adjustment because if a substring is not found to be a match during string matching, it is deemed incorrect. However, the case of deletion is handled differently because of the fact that a keystroke that results in a deletion error can be both correct and incorrect – incorrect for the preceding string and correct for the proceeding string.

**Table 2.** Top 5 Bigrams and Trigrams Extracted from the Corpus

Bigram	Occurences	Trigram	Occurences
th	9709171	the	6076523
he	8552661	ing	3227179
in	7913861	and	2998065
an	6389345	ion	1716878
er	6348583	ent	1519196

After preprocessing, a sliding window decomposes the corpus and keystroke data. The character sequences from the corpus decomposition are referred to as ngrams, with ngrams of length two and three being referred to as bigrams and trigrams, respectively. The sequences attained from the keystroke data are referred to as nstrokes, with nstrokes of length two and three referred to as bistrokes and tristrokes, respectively. Every ngram encountered in the corpus is stored alongside its frequency of occurrence. Whereas each correctly typed nstroke is recorded as a tuple of its characters and a positional vector indicating the keys used to type them, the tuple is a key for storing each unique nstroke’s typing times. For instance, the identifier for the nstroke representing the string ‘the’ on AZERTY, QWERTY, and QWERTZ keyboards would be  $(-1, 3), (1, 2), (-3, 3)$ , ‘the’) and this would be used to store the typing duration of each instance of this pattern regardless of layout.

**Table 3.** Keyboard Character Mapping for QWERTY

Q	W	E	R	T	Y	U	I	O	P
A	S	D	F	G	H	J	K	L	;
Z	X	C	V	B	N	M	,	.	/
SPACE									
$(-5, 3)$	$(-4, 3)$	$(-3, 3)$	$(-2, 3)$	$(-1, 3)$	$(1, 3)$	$(2, 3)$	$(3, 3)$	$(4, 3)$	$(5, 3)$
$(-5, 2)$	$(-4, 2)$	$(-3, 2)$	$(-2, 2)$	$(-1, 2)$	$(1, 2)$	$(2, 2)$	$(3, 2)$	$(4, 2)$	$(5, 2)$
$(-5, 1)$	$(-4, 1)$	$(-3, 1)$	$(-2, 1)$	$(-1, 1)$	$(1, 1)$	$(2, 1)$	$(3, 1)$	$(4, 1)$	$(5, 1)$
$(0, 0)$									

## 2.2 Data Analysis

Our analysis limits the data used to typing sessions from participants who reported using 9 to 10 fingers for typing because they are most likely to adhere to touch typing practices. Touch typing is a standardized method where typists use muscle memory to locate keys without looking at the keyboard, and it is the most widely used keyboard input methodology. Although, in practice, users may deviate from perfect touch typing, this approach is chosen because it enables a systematic examination of typing patterns while still reflecting the effect of shortcuts the participants take.

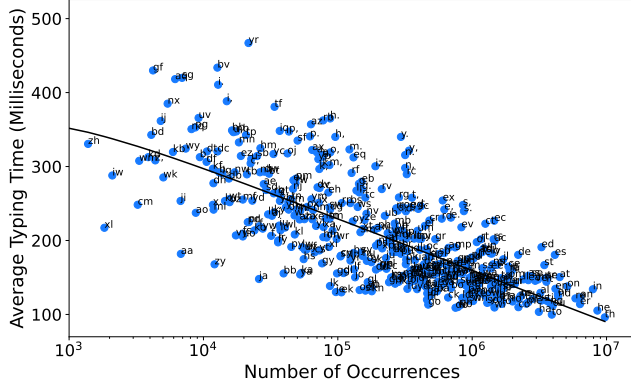
### 2.2.1 Bistroke Analysis

The bistroke analysis reveals that frequency is the most significant predictor of typing time, and as seen in figure 1 the typing speed follows a logarithmic curve with frequency, demonstrating that bistrokes with a high frequency are aided by muscle memory regardless of placement. After accounting for the effect of a bistroke’s frequency, we can determine the influence of key placement and positional relationships to other keys. Naturally, the analysis reveals that the relative dexterity of each finger plays a role in typing keys assigned to that finger, with pinkies being far slower than the other fingers and the middle and index being faster. Interestingly, the top row is predicted to be faster than the home row by a marginal amount, which realistically may be the case for QWERTY users as an adaptive behaviour. This likely represents a bias in our data. An analysis of positional relationships reveals three major categories of bistroke:

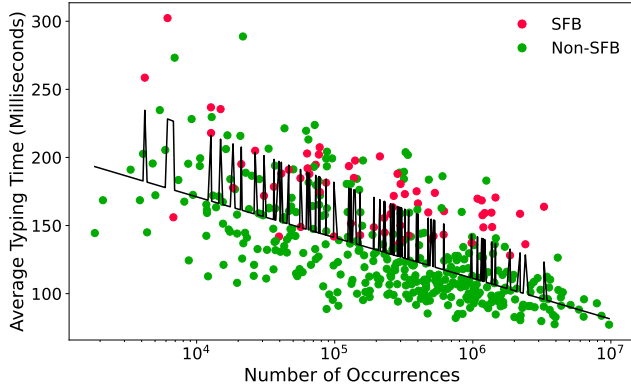
- **ALT (Alternating Bistroke):** a bigram typed on both hands by alternating sides of the keyboard. This is the fastest category.
- **SHB (Same-hand Bistroke):** a bigram typed with the same hand. This category is typically slower due to the need for sequential finger movements on the same hand and, therefore, greater coordination.
- **SFB (single-finger Bistroke):** A bigram typed using the same finger twice. This is the slowest category, because the keys cannot be hit in rapid succession. It is especially slow for data at high words per minute.



An important takeaway from this analysis is that how much a feature contributes to the speed prediction changes depending on the WPM range of the data. Therefore, our proposed keyboard optimization methodology considers this and optimizes for specified ranges.



**Figure 1.** Relationship Between English Bigram Frequency and Typing Time on QWERTY



**Figure 2.** The Effect of SFBs and Frequency on Typing Time (> 80WPM)

### 2.2.2 Tristroke Analysis

Several features unique to tristrokes are identified but are not found to be very statistically significant for the cost functions defined, so we choose not to include them to avoid overfitting. The exception for this is the single-finger-skipstroke category. Like the single-finger bistroke, it is two strokes sharing a finger, but instead, it is separated by one stroke in between. These are important to analyze as for fast typists, single-finger skipstrokes behave more like delayed single-finger bistrokes. Other tristroke features merit future exploration.

### 2.3 Creating the Objective Function

The goal of the objective function is to, for a specific WPM range, predict the typing time of a corpus given a layout. To achieve this, we create a cost function that evaluates each trigram in the corpus, predicts its typing time multiplied by its occurrence count, and takes the sum of the results. Using a manually devised cost function from our findings instead of using a neural network or similar methods, we prevent overfitting on the limited dataset. From here, we use the Levenberg-Marquardt algorithm to fine-tune the free parameters in the cost function to best fit the data. Each cost function outlined below was curated to describe the trends outlined in the data analysis while keeping the number of parameters minimal to ensure generalizability.

To derive the tristroke cost function, we must first derive a bistroke cost function. The bistroke function, denoted as  $C(b)$ , takes a bistroke  $b$  of category  $i = \{1, 2, 3\}$ , with base positional penalties  $P_x$  and  $P_y$ , categorical positional penalties  $P_x^{(i)}$  and  $P_y^{(i)}$ , and a frequency penalty  $f(b)$  and predicts the time to type it. Following our analysis, the frequency penalty is put into a logarithmic function with the free parameters  $p_1$ ,  $p_2$  and  $p_3$ . The columnar penalties  $P_x$  and  $P_x^{(i)}$  are determined by taking the absolute value of the x-coordinate from the second key in the bistroke. Each possible column value is assigned a free parameter to weigh its significance.

Similarly, row penalties are determined based on their y-coordinate, with each possible y-coordinate being assigned a free-parameter. We can safely assume that most participants use a row-staggered layout. On a row-staggered keyboard, the top row has a  $-0.25$  key offset and the bottom row has a  $0.5$  key offset. For single-finger bistrokes, we introduce a distance parameter  $\Delta$  to represent the row-stagger adjusted distance between keys; for other cases,  $\Delta$  defaults to 1. We add a free-parameter  $p_4$  to the distance to establish a baseline penalty weight that determines the amount that distance contributes on top of this effect and to prevent  $\Delta$  from becoming zero and negating the placement penalties. Finally, we set the bistroke cost function to the product of the finger penalties and the frequency logarithm to amplify the impact of features across different frequency ranges:

$$C(b) = (p_0 \log(f(b) + p_1) + p_2) \times \left(1 + P_x(b)P_y(b) + P_x^{(i)}(b)P_y^{(i)}(b)(\Delta + p_4)\right)$$

For the tristroke cost function  $C(t)$ , we take the sum of the cost of its constituent bistrokes  $b_1(t)$  and  $b_2(t)$ , then add a small penalty for the trigrams associated skipstroke  $s(t)$ , if it is a single-finger skipstroke. The single-finger skipstroke penalty, like the single-finger bistroke penalty, considers positional penalties  $P_x$  and  $P_y$  as well as a distance penalty  $(\Delta + p_5)$ . It should be noted that the single-finger skipstroke

penalty defaults to 0 if there is no single-finger skipstroke.

$$C(t) = C(b_1(t)) + C(b_2(t)) + P_x(s(t))P_y(s(t))(\Delta + p_5)$$

Finally, the data is limited to a desired WPM range. The bistroke cost function is then fit to the bistroke data, and the tristroke cost function is fit to the tristroke data. The MAE of the fit changes depending on the WPM, but the  $R^2$  metric stays the same. For 80WPM, this results in  $C(b)$  having an MAE of 12 milliseconds and an  $R^2$  of 0.78.  $C(t)$  is intentionally made less accurate to not overfit with an MAE of 26 milliseconds and an  $R^2$  of 0.42. The cumulative cost  $C(l)$  for a layout  $l$  is the sum of each typable tristroke’s predicted typing time multiplied by its number of occurrences, effectively estimating the typing time for a given layout across the corpus:

$$C(l) = \sum_{t \in T} (C(t) \times f(t))$$

this is the objective function passed onto the simulated annealing algorithm for optimization.

## 2.4 Simulated Annealing

*Simulated annealing* is a metaheuristic optimization technique inspired by the annealing process in metallurgy, where metals are cooled gradually to achieve a stable state with minimized energy [8]. In the context of keyboard optimization, it is used as an iterative algorithm that explores different keyboard layouts by gradually accepting key swaps that reduce the overall energy (i.e. predicted typing time). Unlike gradient descent and other methods that always aim for an absolute minimum, simulated annealing allows for occasional acceptance of higher-cost solutions to explore a broader search space, preventing the algorithm from getting stuck in local optima. Eventually, simulated annealing converges towards a stable, low-cost configuration – the global minimum or an approximation of it.

Selecting the parameters in simulated annealing for convergence can be challenging, usually requiring experimentation or domain knowledge [2]. We outline a dynamic approach to setting the initial temperature and termination criterion to address this. This ensures layouts can be explored across a variety of data modifications. For instance, we may want to analyze layouts at different ranges of words per minute (WPM), or we may only want to optimize for a subset of the total keys. For such cases, the cost function may evaluate to drastically different values, and the number of swaps necessary to converge on a layout may change, necessitating different parameters.

### 2.4.1 Transitions

The first step in implementing simulated annealing is defining a comprehensive set of solutions that represent potential configurations for the optimization. It is essential that the solution space is consistent and that changes are gradual enough to allow simulated annealing to converge. For keyboard optimization, we establish a solution to be a mapping

between keys and positions on a matrix as seen in table 3. The data structure used to map keys and positions consists of two hashmaps. One hashmap maps coordinates to keys and the other keys to coordinates. By using hashmaps, the two operations that need to be performed on the matrix, look-ups, and swapping, can be done in constant time. Furthermore, the coordinate system devised helps encapsulate real-world usage of the keyboard by encoding position in a way that is descriptive of relevant relationships: fingers are grouped by absolute value, the hand is determined by sign, and distance is easily calculated. The neighborhood of a given layout consists of all layouts that can be generated as the result of a single key pair swap.

The probabilistic criterion governing the acceptance of candidate solutions in the simulated annealing algorithm is derived from the Boltzmann distribution function, a fundamental concept in statistical mechanics used to describe the likelihood of particles inhabiting various energy states within a system [7]. This criterion is mathematically represented as follows:

$$A(i) = \begin{cases} \exp\left(-\frac{\Delta E}{T}\right), & \text{if } \Delta E > 0 \\ 1, & \text{otherwise} \end{cases}$$

Here,  $\Delta E$  represents the change in energy within the system or, in this case, in the estimated typing time between the current layout and the candidate layout, and  $T$  denotes the temperature parameter of the system. If the change in typing time is negative ( $\Delta E < 0$ ), indicating an improvement in estimated typing time, the candidate layout is accepted with certainty. However, if the change in estimated typing time is positive, indicating a worsening layout, the candidate layout is accepted with a probability determined by the Boltzmann factor, which is more likely to accept slight decreases in performance over large ones.

This choice of probabilistic criterion allows the simulated annealing algorithm to converge on an optimal or near-optimal solution through gradual refinement [6]. Initially, the temperature is high, allowing the algorithm to explore the solution space and escape local minima by occasionally accepting suboptimal moves. As the algorithm progresses and the temperature decreases, the probability of accepting suboptimal moves diminishes. The reduction in temperature is controlled by a cooling schedule, which dictates how fast and effectively the algorithm converges to a solution. We adopt a monotonically decreasing geometric cooling schedule, defined as follows:

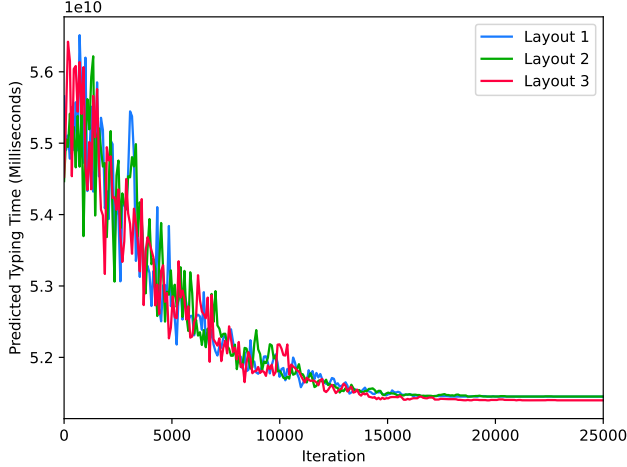
$$T_i = a^i T_0$$

where  $T_0$  is the initial-temperature and  $a$  is the cooling-rate,  $0 < a < 1$ .

### 2.4.2 Setting the Initial Temperature

Before any transitions are performed, an initial temperature must be set for the cooling schedule to iterate on. A





**Figure 3.** Convergence of Estimated Typing Time during Simulated Annealing

common approach to setting the initial temperature is manually estimating the initial temperature based on the problem domain and the characteristics. This initial temperature is frequently chosen so that the acceptance probability at the beginning of simulated annealing is approximately equal to some value. Lacking relevant literature for our problem domain, we resort to the approach outlined in Ben-Ameur [2]. We set a variable  $\chi_0$  to be the desired initial acceptance probability of suboptimal transitions at the beginning of the algorithm. An iterative method is employed to compute the initial temperature  $T_0$  such that the acceptance probability approaches  $\chi_0$ . To compute the acceptance probability  $\hat{\chi}(T)$  for a given  $T$ , first, we randomly generate a set  $S$  of positive transitions (i.e., transitions where  $E_{before} < E_{after}$ ), where each transition  $s$  represents a swap of two keys. To derive the final conditional expectation for accepting a positive transition, we take the ratio between the probability of accepting a positive transition  $s$  and the probability of generating the transition  $s$ :

$$\hat{\chi}(T_n) = \frac{\sum_{s \in S} \exp\left(-\frac{E_{after}(s)}{T_n}\right)}{\sum_{s \in S} \exp\left(-\frac{E_{before}(s)}{T_n}\right)}$$

Then, to find  $T_0$ , we define an initial guess temperature  $T_1$ , and use a recursive formula until  $|\hat{\chi}(T_n) - \chi_0| \leq \epsilon$  for some error  $\epsilon$ :

$$T_{n+1} = T_n \left( \frac{\ln(\hat{\chi}(T_n))}{\ln(\chi_0)} \right)$$

#### 2.4.3 Termination Criterion

Finally, we establish the termination criterion for the simulated annealing process to be the point at which the number of iterations lacking improvement reaches the probabilistic threshold that all potential swaps have been evaluated. We

do this to ensure the simulated annealing algorithm has settled on some minima while allowing sub-optimal swaps to be heuristically accepted, potentially escaping local minima.

To find the termination criterion, let  $S$  be the number of iterations to evaluate all potential swaps. Let  $k$  be the number of keys, for  $k$  keys, the number of swappable key pairs  $n = \binom{k}{2}$ . We consider  $S = \sum_{i=1}^n s_i$ , where  $s_i$  is the number of iterations required to evaluate the  $i$ -th pair after  $i-1$  pairs have been evaluated.

The probability  $P_i$  of selecting a new pair to swap is  $P_i = \frac{n-(i-1)}{n} = \frac{n-i+1}{n}$ . Consequently,  $s_i$  follows a geometric distribution with an expectation of  $E(s_i) = \frac{n}{n-i+1}$ . By the linearity of expectations, we derive:

$$E(S) = n \sum_{i=1}^n \frac{1}{i}$$

To increase performance, we use the approximation:

$$E(S) = n \log n + \gamma n + \frac{1}{2} + O\left(\frac{1}{n}\right)$$

where  $\gamma \approx 0.5772156649$  is the Euler-Mascheroni constant. Since the number of iterations must be a positive integer, we set our final stopping point to be:

$$\left\lceil \binom{k}{2} \log \binom{k}{2} + \gamma \binom{k}{2} + \frac{1}{2} \right\rceil$$

The layout produced in this paper optimizes for the main block of 30 keys. So the number of possible swaps is  $\binom{30}{2} = 435$  resulting in a stopping point of  $\left\lceil \binom{30}{2} \log \binom{30}{2} + \gamma \binom{30}{2} + \frac{1}{2} \right\rceil = 2,895$  iterations lacking improvement.

## 3 Results

For the final layout, we choose to use the 30 most common characters in our corpus, which is the alphabet, plus the addition of four special characters: the comma, period, hyphen, and apostrophe. This layout differs by 2 characters from the QWERTY layout, so to compare the final layout with QWERTY requires we only predict the time for strings in the corpus that can be typed on both layouts. Although, the average typing speed was 47 WPM, we set the target WPM to  $\geq 80$  as a litmus test for proficiency. This results in single-finger bistrokes having a greater impact and frequency having less of an impact. Optimizing for an above average WPM like this is desirable if the goal is to raise the upper threshold of potential typing speed. Using the methodology in this paper, one could also aim to optimize for the average user, improving the mean speed. The resulting layout can be seen in table 4.

The predicted time to type the iWeb corpus on QWERTY is 54,934,581,797 milliseconds, dvorak is 52,565,248,833 milliseconds, a speed up of 4%, and the layout produced in this paper is predicted to take 51,429,827,290 milliseconds, a speed up of 6%. Our analysis revealed positional categories of bistroke that played a significant role in the prediction of

**Table 4.** Generated Layout for Typing Speeds  $\geq 80$  WPM

M	R	T	C	W	,	K	A	E	'
L	N	D	S	V	Y	U	O	I	G
H	X	Z	B	F	.	-	Q	J	P
SPACE									

typing speed. Namely, ALTs which are shown to be faster, and SFBs which are shown to be slower. For QWERTY, 18.3% of all bistrokes are ALTs and 5.7% of them are SFBs, dvorak has 33.6% ALTs and 2.8% SFBs, and the layout produced in this paper has the best result of 33.6% ALTs and 1.4% SFBs. The fact that speed optimization yields only marginal improvements in typing efficiency is not unexpected, but remains a valuable take away as it suggests prioritizing theoretical features over solely focusing on speed optimization may offer more substantial benefits.

#### 4 Future Work

Future research lies in addressing the biases introduced by the similarities of the QWERTY, AZERTY, and QWERTZ layouts, since they hinder the objective function's ability to generalize. To facilitate this future research, an open-source tool called Kiakl was developed to crowd-source data from alternative keyboard layouts. Further avenues of future investigation include exploring the inclusion of more optimization criteria, such as finger dexterity and speed as proxies for comfort, as well as investigating more efficient variations of the simulated annealing algorithm.

#### References

- [1] TK Amell and S Kumar. 2000. Cumulative trauma disorders and keyboarding work. *International journal of industrial ergonomics* 25, 1 (2000), 69–78.
- [2] Walid Ben-Ameur. 2004. Computing the Initial Temperature of Simulated Annealing. *Computational Optimization and Applications* 29, 3 (01 Dec 2004), 369–385. <https://doi.org/10.1023/B:COAP.0000044187.23143.bd>
- [3] Mark Davies. 2018. The iWeb Corpus. Available online at <https://www.english-corpora.org/iWeb/>.
- [4] Vivek Dhakal, Anna Feit, Per Ola Kristensson, and Antti Oulasvirta. 2018. Observations on Typing from 136 Million Keystrokes. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM. <https://doi.org/10.1145/3173574.3174220>
- [5] Jan Eggers, Dominique Feillet, Steffen Kehl, Marc Oliver Wagner, and Bernard Yannou. 2003. Optimization of the keyboard arrangement problem using an ant colony algorithm. *European Journal of Operational Research* 148, 3 (2003), 672–686.
- [6] Vincent Granville, Mirko Krivánek, and J-P Rasson. 1994. Simulated annealing: A proof of convergence. *IEEE transactions on pattern analysis and machine intelligence* 16, 6 (1994), 652–656.
- [7] Stewart Harris. 2004. *An introduction to the theory of the Boltzmann equation*. Courier Corporation.
- [8] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science* 220, 4598 (1983), 671–680.
- [9] Lorie M Liebrock. 2005. *Proceedings of the 2005 ACM symposium on Applied computing*. ACM.
- [10] Lissa W Light and Peter G Anderson. 1993. Typewriter keyboards via simulated annealing. *AI Expert* (1993).
- [11] Gonzalo Navarro. 2001. A guided tour to approximate string matching. *ACM computing surveys (CSUR)* 33, 1 (2001), 31–88.
- [12] Marc Oliver Wagner, Bernard Yannou, Steffen Kehl, Dominique Feillet, and Jan Eggers. 2003. Ergonomic modelling and optimization of the keyboard arrangement with an ant colony algorithm. *Journal of Engineering Design* 14, 2 (2003), 187–208.
- [13] Koichi Yasuoka and Motoko Yasuoka. 2011. On the Prehistory of QWERTY. *Zinbun* 42 (2011), 161–174.
- [14] Peng-Yeng Yin and En-Ping Su. 2011. Cyber Swarm optimization for general keyboard arrangement problem. *International Journal of Industrial Ergonomics* 41, 1 (2011), 43–52.

**Disclaimer: The contents of this paper are reproduced from the 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), held at the University of Nevada, Las Vegas, USA, from January 8th to 10th, 2024.**

# Elephant: LLM System for Accurate Recantations

Praise Chinedu-Eneh  
Department of Computer Science  
Winona State University  
Winona, MN 55987, USA  
praiseneh@icloud.com

Trung T. Nguyen  
Department of Computer Science  
Winona State University  
Winona, MN 55987, USA  
trung.nguyen@winona.edu

**Abstract**—This paper investigates the crossroads between Large Language Models (LLMs), Information Retrieval (IR), and the growing issue of misinformation in this current age of the Internet. Large Language Models, including the GPT and LLaMA series (both implemented in this paper), have given us profound insight into how we interact on the Internet and in real life. They have also innovated the way we interact with information. However, these innovations also pose significant shortfalls, particularly in the domain of misinformation from the text. The main contribution of this paper lies in developing a proposed strategy to mitigate the risks of misinformation seen on the Internet and generated from LLMs with a focus on public individuals by recanting statements they have made and the retrieval of said statements. We propose a multi-faceted approach that includes utilizing GPT3.5 and the open source LLaMA2 LLMs, finetuning data curation, and integrating accuracy mechanisms to ensure the most relevant and accurate information is retrieved. The efficacy of this methodology is measured using a cosine similarity metric. Considering that the recanting of this model must be at or as close to the original statement as possible, this metric is deemed most fitting. Findings later in this paper deemed a similarity recall of 90.42% on average with the GPT3.5 variant and 88.29% on average with the LLaMA2 variant, both in zero-shot examples, indicating the core semantic meanings were retrieved with variations on the format of illustration.

**Index Terms**—Information Retrieval, LLMs, Information Recantations, Misinformation, Machine Learning, Artificial Intelligence

## I. INTRODUCTION

The rapid evolution of Large Language Models (LLMs) like GPT and LLaMA series has significantly impacted how we interact with and retrieve information online. This paper delves into the intersection of LLMs and Information Retrieval (IR), focusing on addressing the challenge of misinformation prevalent in the digital age.

### A. Background and Evolution

We begin by exploring the development of traditional text information retrieval methods and their transformation with the advent of LLMs. The progression from early hypothetical computerized systems in the mid-20th century, such as the "Memex" concept by Vannevar Bush [1], to modern IR algorithms highlights a significant shift towards more sophisticated techniques. This shift is further accentuated by introducing LLMs, which bring an unprecedented ability to understand context, generate coherent responses, and process vast datasets.

The evolution of search engines, particularly with the emergence of Google and its PageRank algorithm [2], marks a pivotal shift in IR. This era saw the application of advanced NLP techniques and graph-based algorithms, enhancing the relevance and contextuality of search results.

A brief overview of the neural network architecture that underpins LLMs sets the stage for understanding their capabilities and limitations. These models, mirroring aspects of the brain's neural network, have revolutionized data processing and retrieval and introduced challenges, especially in misinformation.

### B. Proposed Approach

This work mainly contributes to developing a strategy to mitigate misinformation risks by recanting statements made within multiple broad documents, utilizing GPT3.5 and LLaMA2 LLMs. This includes fine-tuning data curation and integrating accuracy mechanisms, assessed using a cosine similarity metric to ensure precise and reliable information retrieval.

The paper aims to offer a comprehensive analysis of LLMs in the context of IR and misinformation, providing insights and methodologies to leverage these models responsibly in our increasingly digital society.

In summary, our contributions are listed below:

- A multi-faceted information retrieval approach utilizing GPT3.5 and LLaMA2 models, including fine-tuning LLM data curation and topic modeling steps.
- A comprehensive study of accuracy mechanisms using cosine similarity and topic verification to ensure the most relevant and accurate information is retrieved.
- A proposed strategy to mitigate the risk of misinformation generated by public individuals and online statements.

## II. RELATED WORKS

This section reviews significant literature on Information Retrieval (IR) and Natural Language Processing (NLP), particularly focusing on the integration of modern NLP techniques with IR systems and the use of Large Language Models (LLMs).

### A. Early Approaches in IR

Gerald Salton (1971) [3] explored the System for the Mechanical Analysis and Retrieval of Text (SMART), which

allowed for documents to be represented as vectors in a multi-dimensional space. This key feature changed the landscape of text retrieval. Karen Spärck Jones (1990) [4] discussed the role of artificial intelligence in IR, marking a significant development in the field. Karen Spärck Jones (1997) [5] addressed the role of linguistically motivated indexing (LMI) in text retrieval. The paper reviewed historical and contemporary research in automated LMI via NLP, concluding that while LMI is not necessary for effective retrieval, it plays vital roles within broader information-selection systems.

### B. Traditional ML Techniques in IR

1) *Information Retrieval Research and Digital Libraries:* Karen Spärck Jones (2006) [6] provided insights into the history of IR research, emphasizing the value of statistical techniques. The paper argued for the potential of these methods in the evolving landscape of digital libraries, which deal with heterogeneous, large-scale, and dynamic content.

2) *Natural Language Processing in IR:* Matthew Lease (2007) [7] explored methods for integrating modern NLP with state-of-the-art IR techniques and emphasized the application to conversational speech data. This research underlined unique challenges in speech data retrieval compared to text.

3) *Large-Scale IR in Software Engineering:* Michael Unterkalmsteiner et al. (2023) [8] described applying IR techniques in industrial-scale software engineering. Through an empirical study, they illustrated the practical application and evaluation of IR methods in test case selection.

4) *Machine Learning in Dynamic IR:* Sharon Jiang et al. (2023) [9] conceptualized machine learning for dynamic retrieval of Electronic Health Record notes. Their work highlighted the potential of EHR audit logs as a data source for training models in clinical contexts.

### C. Deep Learning and LLMs for IR

1) *Large Language Models in IR:* LLaMA 2 [10] and GPT 3.5 [11] mentioned throughout this paper are large language models (LLM). These machine learning statistical models make predictions on the next word to generate based on the previous scope of words and its own training data. Yutao Zhu et al. (2023) [12] provided a comprehensive survey on integrating LLMs in IR systems. The survey covered aspects like query rewriters, retrievers, rerankers, and readers, offering insights into the evolving role of LLMs in IR. Roulin Peng et al. (2023) [13] explored embedding-based retrieval using domain-agnostic LLMs for extracting structured data from agricultural documents, highlighting the potential of LLMs in domain-specific applications. Yan Gong and Georgina Cosma (2023) [14] presented Boon, a cross-modal search engine that integrates the GPT-3.5-turbo LLM with the VSE network VITR for enhanced image-based information retrieval. Chinese IR Community (2023) [15] summarized the outcomes of a strategic workshop, discussed the impact of LLMs on IR research, and proposed new technical paradigms.

2) *Toolkits and Synthetic Data Generation in IR:* Hugo Abonizio et al. (2023) [16] introduced the InPars Toolkit for synthetic data generation in neural IR research and later released InParse-v2 [17]. The toolkit provided a unified and reproducible pipeline for synthetic data generation, encompassing generation, filtering, training, and evaluation processes.

## III. PROPOSED METHODOLOGY

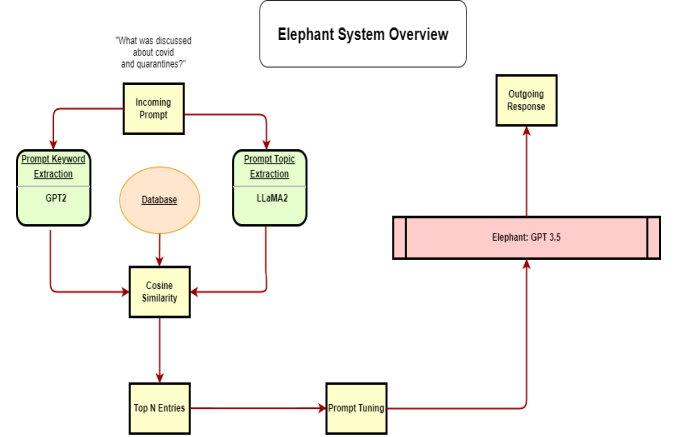


Fig. 1: An Overview of the Proposed Elephant System

This section will present our proposed methodological framework, the Elephant System.

### A. Data Collection

This paper aims to establish a methodological framework that allows LLM to accurately recount statements made within a text document. We have seen a shift in habitual media consumption from traditional print publications in recent years towards digital mobile devices, particularly video content on social media platforms. In alignment with these requirements, the dataset chosen for fine-tuning the LLM comprised YouTube video transcripts. Specifically, a series of podcasts from All-In with Chamath Palihapitiya, Jason Calacanis, David Sacks, and David Friedburg [18] were selected as the primary data source. The rationale behind selecting this data source stems from the long-form nature of podcasts. This inherently allows the data source to include a plethora of statements on various topics over an extended duration. These data characteristics will present a diverse array of topics challenging the LLM, requiring the LLM to maintain accuracy and consistency across wide ranging subjects over lengthy discourse. The dataset comprises transcripts from multiple podcast episodes. This is done to ensure a broad representation of topics, requiring the model to not only recant statements from a topic within an episode but also related topics across differing episodes. Given that this research serves as a proof of concept, the dataset is static rather than dynamic. The subset provides a sufficient snapshot of the series, enabling an adequate evaluation of the LLM's performance in recantation tasks.

## B. Elephant

Our proposed system, Elephant, is implemented in two variants, LLaMA2-7B [10] and GPT 3.5 [11] models, both designed with the capability for precise textual recall, as the name suggests. This model demonstrates a robust ability to accurately replicate statements from text documents, despite the diverse and varied dataset from which it's required to recall. In practice, Elephant can sift through large volumes of text and echo the original content with minimal deviation, particularly useful in applications where verbatim recall is necessary.

The illustration above, Fig. 1, illustrates a holistic representation of the proposed Elephant system. The diagram presents four primary entities and three processing stages: the prompt keyword extraction component, the prompt topic extraction component, the database, and Elephant, the main LLaMA 2 or GPT 3.5 based model. Intermediate steps include the cosine similarity measure, curation of the top N entities, and the prompt tuning step. Starting with the incoming prompt, which can be something such as 'What was discussed about covid and quarantines', the diagram traces the workflow through two phases, topic extraction, keyword extraction, and database extraction, culminating at the core of the system—Elephant, which is based upon either the LLaMA2 or GPT 3.5 architecture. The intermediary steps, including the application of cosine similarity measures, the selection of the top N entities, and the final prompt tuning stage, are also mapped to provide insight into the system's processing pipeline. Essentially, the system will accept a user query prompt, assess the keywords and topics of the prompt, retrieve and select further relevant information on how to properly comply with the request, use that information to generate text from the core model, and then respond to the user. Applying this to statements provides an easily accessible way to perform a look-up on public individuals, allowing a more holistic assessment of whether supporting the public individual aligns with their own ideas.

## C. Prompt Deconstruction

Considering the diagram illustrated in Figure 1, examining the individual components and their responsibilities within the overall architecture is pertinent.

We can group the larger system into three smaller subsystems of individual components. This section will focus mainly on analyzing the first subsystem, Prompt Deconstruction. Delineated by Fig. 2, prompt deconstruction is initialized by an incoming prompt. The nature of these kinds of prompts will be user queries. The overarching system, especially this portion of the system, is predicated on the assumption that the user is querying based upon prior knowledge of the general nature of the subject entity the corpus is composed of. This means the user will understand that the resultant text will be a culmination of statements made by a particular entity, whether an individual or an establishment. In this paper, the entities in question are the proprietors of the podcast. Although functionality could be extended to retrieve statements amongst an array of entities, that is not the focus of this paper.

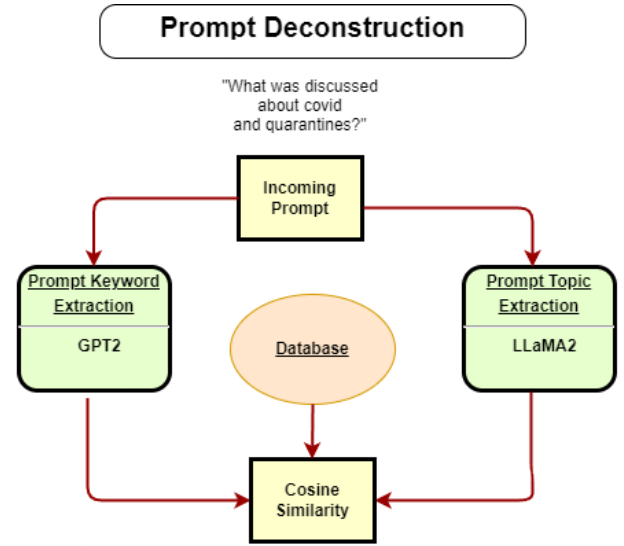


Fig. 2: A close up of the prompt construction step

1) *Topic Extraction*: The user prompt, once accepted, is taken and deconstructed bidirectionally and then fed into a topic extraction and a keyword extraction system, respectively. The former, topic extraction, is done using a base LLaMA2-7B model. Although other topic modeling approaches, such as Latent Dirichlet Allocation (LDA) using tools like spaCy [19] or Gensim [20], have been known to provide sufficient topic words, the backbone of these approaches still tends to rely on metrics adjacent to term frequency. This can lead to including topic words with little to no meaning or relevance compared to the scope of the episode entry from which it is derived. For this reason, we found that using these packages in a domain applicable to keyword extraction tends to produce higher functioning results, which is mentioned later in this paper, and using transformer models such as LLaMA2 or the GPT series leads to better topic extraction. Due to the inherent attention mechanism factor within transformer models, they can derive semantic meaning with a higher degree of relevance. The downside to such an approach is the embodiment of the main contribution of this paper, the inconsistencies in the output topic words extracted.

"Extract 2 topic words from the following text on new lines  
-- text --  
Then return it as a numbered list with a period after each number  
DO NOT respond with brackets or parenthesis: "

Fig. 3: An example of topic extraction prompt

This is mitigated via a three-step approach illustrated in Fig. 3. First, at the time of inference, the model parameters are set to result in a more deterministic output. Setting the temperature output to an incredibly low floating-point number, in this case 0.1, is a main contributor. Second, the prompt passed through to the topic extraction model. This must be precise in what and how the topics are extracted from the prompt



using the model. Attributes such as explicitly requesting the number of topics, the composition of the topics—words, phrases, or otherwise—and the format to provide them are all necessary. To limit deviation, it is useful to explicitly state what is not permissible to respond with. In this case, we specifically requested the omission of brackets and parenthesis when listing topics. Third, the nature of the embeddings being compared will also mitigate deviation. This will be detailed during the cosine similarity step.

2) *Keyword Extraction*: The latter, keyword extraction, uses a multi-step process where the aforementioned spaCy and Gensim packages are applied. Initially, the spaCy library is imported and loads a natural language processing model through which we pass the prompt. This will return a document representation comprised of words parsed as text. A loop is initiated on the document in which for each token in the document if the token is not among the set of words deemed as ‘stop words’, these include words such as ‘the’ and ‘is’, it will be added to a list. We then use Gensim’s TextRank [20] keyword package to extract the top N potential keywords. Then, for each token, we generate the GPT2 embeddings, and then the average embedding of all the tokens is calculated to represent the overall semantic meaning of the text. We then generate embeddings for each extracted keyword and calculate the cosine similarity between the average text embedding and each keyword. Finally, keywords with a cosine similarity above a certain threshold are considered significant and appended to the final list.

```
(
  "_id": {
    "$oid": "string"
  },
  "uID": "string",
  "refID": "string",
  "entryDate": "string",
  "transcriptStart": "string",
  "sample": {
    "uID": "string",
    "promptInstruct": "string",
  },
  "topics": [
    "string",
    "string",
    ...
  ],
  "keywords": [
    "string",
    "string",
    ...
  ]
)
```

Fig. 4: An example of the database schema

3) *Database*: For this application, a Mongo Database was used. The composition schema of each episode entry, referenced in Fig. 4, is as follows: a reference id used to reference which episode the entry came from. An entry date is used as an indicator as to when the entry was added to the database. Also, the transcript start property within the data is used to identify at what point in the episode we are in. A sample object is a property within the data, including the actual text transcript. Finally, a topics array is within the data, indicating the dominant topics associated with the episode entry, and a keywords array contains the keywords of the episode entry.

4) *Cosine Similarity*: A critical step-component of the Elephant system is its capability to ascertain the degree of resemblance between the incoming prompt and the vast number

of entries within its database. To accomplish this, the system leverages a computed cosine similarity metric, a quantitative measure of the cosine angle between two embedding vectors in a multi-dimensional space, reflecting how closely aligned the compared terms are in content and meaning.

Each transcript entry is converted into an embedding, essentially a numerical representation that holds data, such as the semantic meaning of the text, which is then compared against the prompt embedding. Using PyTorch’s cosine similarity function [21], we can leverage its efficiency to calculate cosine similarity between embeddings. Furthermore, an additional layer of comparing the topics and keywords of the prompt with the entries in the database is also applied, producing a similarity score, referred to as attribute similarity. In essence, both an embedding-based similarity and an attribute-based similarity are applied and are considered to indicate a quantitative measure of the overall resemblance to the prompt. Subsequently, a combined similarity is calculated to create a holistic similarity metric. This is done by averaging the embedding similarity and the attribute similarity and is meant to represent how closely an entry matches the incoming prompt’s semantic content and thematic attributes.

#### D. Similarity Selection

This section overviews the next subsystem within the larger Elephant system. It will focus mainly on selecting entries with the highest similarity score and how they are tuned before presenting them to the main Elephant model.

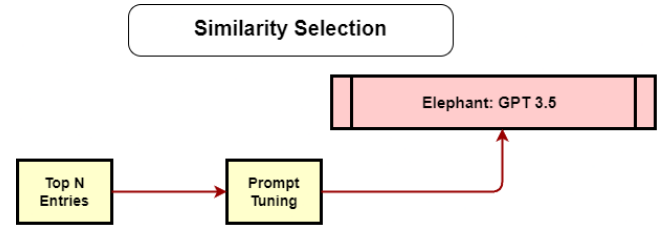


Fig. 5: An overview of Similarity Selection process

1) *Top N Entries*: After the computation and comparison of cosine similarities, the scores are organized in descending order, prioritizing entries with the highest levels of similarity. The similarity values are floating point numbers ranging from -1 to 1, with a value of 1 denoting perfect similarity. As illustrated in Figure 3, this process culminates in selecting the top N entries. This critical step hones the model’s focus on the content most relevant to the given prompt, effectively serving as a filter to enhance the precision of the model’s outputs.

Selecting the top N entries is instrumental in preventing the model from being encumbered by an overabundance of data, which could potentially dilute its attention capacity. Given the architectural design of LLMs, there is a propensity for the model’s attention to decay when processing excessive amounts of information. The concept of attention refers to the window of context the model can reflect upon within the sequence or sequences of text. As the sequence gets longer, information can slip in and out of this window based on a

TABLE I: A representation of topics and cosine similarity

Cosine Similarity (%)	Topics
71.61% (0.4321)	facebook, google, microsoft, apple amazon, twitter, peter thiel, mark zukerberg, trump
65.53% (0.3105)	twitter, death, san francisco, google, android, chief, product office, mobile coin, angel investor, figma
65.01% (0.3002)	girls, pain, twitter, phone monitoring, launch startup science sores boys
64.73% (0.2945)	real estate, florida, facebook, censorship, alternative, competition, platform, management, excuse

criterion, causing the model to lose track of parts of the text, known as attention decay. This decay can lead to less coherent responses and, in some cases, generate responses that deviate from factual accuracy, a phenomenon commonly referred to as ‘hallucination’ [22] within the field. By strategically limiting the scope of data to the most relevant entries, we mitigate the risk of such degradation, thereby preserving the integrity and legibility of the model responses.

The selection of the top N entries, upon face value, may seem trivial, but after closer inspection, this step is also crucial. Table I depicts a selection of a few topics and their associated cosine similarity after the initial prompt of ‘Did they ever discuss Apple or the iPhone?’ In this study, we selected the top N entries based on some arbitrary value, but in post evaluation, we saw that the threshold of the relevance of the topics when associated with cosine similarity is about 0.31. We omitted the intermediary values for the sake of space, but as the table represents, the topic of apple, still presents at a similarity score of 0.4321, then around 0.3105, the topic of Apple specifically is not present, but adjacent topics such as Twitter, Google and San Francisco still exist. As we go below 0.31, topics tend to lose representation of the original user query with topics such as Real Estate. Considering this, it may be wise to select the top N entries based on the cosine similarity value rather than some arbitrary value. The consequences of this approach are two-fold. The entries selected would have more relevance to the initial query but at the cost of likely feeding the model with too much incoming data, if the number of entries above the threshold is substantial, negating the entire motive of this curation step.

2) *Prompt Tuning*: The final step, prior to running inference on the Elephant model, is prompt tuning. During this step, the model is presented with the user’s initial prompt and the associated entries it must quote from to respond accurately. The model is set to have a behavioral pattern of a helpful chatbot, ensuring its tone is adequate for human interaction. Like the topic extraction step, it is crucial to have clear and strict guidelines on how and what the model should be and in what format. This would be the area where you would set certain guidelines. In this study, the model was set to ensure it includes the related links it’s quoting from and to urge the user to follow those links for a more comprehensive understanding, as these are just snippets. After these safeguards are in place,

the text is forwarded to the main inference model.

#### E. Model Output

The final subsystem of our proposed system, as depicted in Fig. 6, is either a finetuned GPT 3.5 turbo model [23] or a fine-tuned LLaMA2 model [10], specifically for accurate information retrieval of statements made within a text corpus. They were finetuned with data similar to an expected question, retrieval, answer conversation format. Designed to select individual quotes, Elephant outputs accurate quotes with the help of the initial prompt tuning and prior steps. The max tokens parameter is set to 1024 tokens. This allows the model to generate many new tokens, although such a limit may never be reached. Every other parameter is using the default value. Alternatively, setting the temperature to a low value, such as 0.2 or 0.3, may also be beneficial.

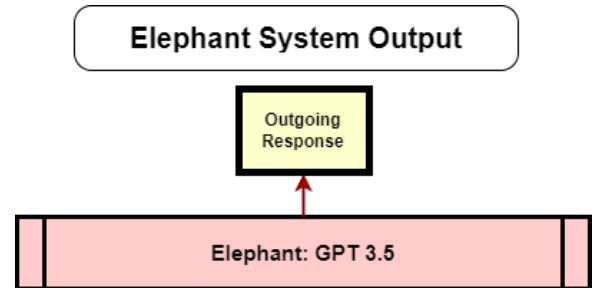


Fig. 6: An overview of the Elephant output

## IV. EXPERIMENTS AND RESULTS

Experiments were carried out on an array of prompts tailored towards the nature of the data being retrieved on the Elephant LLM, with both the GPT and LLaMA2 variants tested. Moreover, the same prompt of ‘Did they ever discuss Apple or the iPhone?’ was run through each model 10 times on differing temperature parameters, and the cosine similarity was calculated by comparison to the reference text embeddings.

#### A. LLaMA2 variant model

Regarding experiments run on the LLaMA2 variant and the nature of this paper, experiments that consider the consistency of the responses and the accuracy of the quotes within them were the focus of evaluation. On each run, the model correctly quoted the entries provided when responding to the user. Unfortunately, the consistency of the model was in question. Figure 5 depicts model responses, with the user prompt stated at the top, and the various prompts listed subsequently as ‘Chatbot’. At a temperature value of 0.6, when a quote is included, the model tends to replace statements with ellipses. This suggests the model may have a propensity to paraphrase, as seen in the first chatbot response, regardless of whether it was explicitly directed not to do so. At a temperature of 0.1, the model tends to include the entire quote, regardless of whether it was told explicitly to do so. Due to the deterministic nature of a 0.1 temperature value, only one response was displayed to avoid redundancy. Both values and the model at large tend to include irrelevant quotes.



User: Did they ever discuss apple or the iphone?  
Temp: 0.6  
Chatbot: Yes, they did! In E64, they mentioned that Apple is no longer seen as a threat and that Microsoft was able to make an offer because of this. They also mentioned in E123 that Apple is a customer of China, and that losing them as a customer would be devastating for China and for Apple as well.  
Chatbot: Sure! Here are some quotes related to apple and the iPhone from the provided transcripts:  
E64: "Microsoft did because microsoft is no longer seen as a threat..."  
E123: "China loses apple as a customer that would be absolutely devastating for them and it would obviously be devastating for apple as well..."  
Temp: 0.1  
Chatbot: Sure! Here are some quotes related to apple and the iPhone from the provided transcripts:  
E64: "Microsoft did because microsoft is no longer seen as a threat...you know they have really been the most untouched in the last decade by antitrust scrutiny."  
E123: "China loses apple as a customer that would be absolutely devastating for them and it would obviously be devastating for apple as well...when we look back on that as a general rubric here do you think we enabled a competitor or we avoided future conflict because of the interdependency?"  
E4: "Facebook's position is no position...we're not gonna police adds...all big five tech companies have said our position is no position."

Fig. 7: The sample responses of LLaMA2 variant model

TABLE II: Performance results of the LLaMA2-based model

Avg (%)	SD	Var	CV	IQR	Temp.
82.04% (0.64070)	0.00127	1.61E-06	0.198%	0	0.1
83.22% (0.66431)	0.04825	0.00233	7.264%	0.04902	0.2
85.32% (0.70644)	0.06932	0.00481	9.813%	0.13126	0.3
85.08% (0.70156)	0.0588	0.00346	8.381%	0.11366	0.4
88.29% (0.76588)	0.03436	0.00118	4.486%	0.04259	0.5

### B. LLaMA2 Results

Table II illustrates the relationship between temperature and cosine similarity, considering the characteristic effect temperature has on the output of the LLM and the overall cosine similarity. As stated throughout this paper, cosine similarity is the primary quantitative performance measure metric.

The columns in Table II are Average (Avg), Standard Deviation (SD), Variance (Var), Coefficient of Variation (CV), Interquartile Range (IQR), and Temperature (Temp), respectively. The data within Table II suggests a trend where the variability in the model responses, as indicated by the standard deviation (SD), increases as the temperature increases, eventually peaking at a temperature of 0.3 and then starting to decrease. Considering that the average cosine similarity steadily increases as temperature rises from 0.1 to 0.5, this suggests that the LLaMA2 variant becomes less deterministic, which aligns with the reference output improvement on average. We can conclude that the LLaMA2 variant tends to produce more varied responses in terms of similarity to the reference as the temperature increases, but only up to a certain point. Beyond that threshold, 0.3, the variability starts to decrease, while the average similarity is independent of this trend and steadily increases as temperature increases, indicating a potential sweet spot around 0.4 to 0.5. This average may be increased by supplying more data during the finetune process.

TABLE III: Performance results of the GPT 3.5-based model

Avg (%)	SD	Var	CV	IR	Temp.
90.42% (0.8083)	0.04054	0.00164	5.016%	0.00901	0.1
90.08% (0.80155)	0.07958	0.00633	9.928%	0.11004	0.2
88.79% (0.77581)	0.05336	0.00285	6.877%	0.09497	0.3
86.76% (0.73528)	0.07015	0.00492	9.540%	0.12543	0.4
86.37% (0.72745)	0.06546	0.00429	8.998%	0.10935	0.5

### C. GPT 3.5

Expectedly, the GPT3.5 variant is evaluated in the same way. Using the same user incoming prompt, the same system message preamble, and being supplied the same transcript data to base its response. Figure 6 depicts the general response of the GPT3.5 variant outputs. The consistency of the model fared much better in listing all the related quotes from the provided transcript data, although in some cases, the model did paraphrase, even when explicitly directed not to, it still maintained the expectation of extracting each related quote. For the sake of brevity, only one response is included, seeing as subsequent responses were more of the same.

1. "them trying to insert this union agenda into anti-trust law you got to think that for example going back to microsoft activation obviously the bankers would have called apple and facebook and google as well i mean it's absurd that they wouldn't have" - E64 (source: [link](youtube.com/watch?v=s9n9db373e8&t=2427.04s))
2. "no pricing you mean facebook and google the consumer doesn't pay so how can there be consumer harm right so clearly that there's that standard of consumer harm doesn't really capture the market power dynamic and then i think the last observation she makes which i think is pretty valid is this idea that you know when facebook bought instagram at that time it didn't seem to be a problem but it definitely allowed them to extend their monopoly from the desktop into mobile and it probably should be scrutinized harder" - E64 (source: [link](youtube.com/watch?v=s9n9db373e8&t=1785.84s))
3. "China loses apple as a customer that would be absolutely devastating for them and it would obviously be devastating for apple as well so when we look back on that as a general rubric here do you think we enabled a competitor or we avoided future conflict because of the interdependency and where do you sit in terms of thinking about engagement versus maybe isolation or something" - E123 (source: [link](youtube.com/watch?v=Zf2tYNhQ5J8&t=3726.299s))

Fig. 8: The sample responses of the GPT3.5 variant model

### D. GPT 3.5 results

According to the findings in Table III, the GPT3.5 variant saw higher variance across the board and a higher average similarity score. This indicates that for each temperature value, the similarity was reasonably high, implying that it tends to reference the correct transcripts but may slightly paraphrase to some degree.

The average cosine similarity trends downwards as the temperature increases, which opposes the previous trend in the LLaMA2 variant, indicating that lower temperatures align better with the reference text. This trend expectedly goes hand-in-hand with the standard deviation as temperature increases, signifying a less deterministic output than expected. The relative spread of data in relation to the mean, as depicted

by the Coefficient of Variation (CV), is higher at temperatures 0.2, 0.4, and 0.5, suggesting the relative variability is greater at these values. As the alignment trends upwards as the temperature decreases and the higher CV values are at a temperature of 0.2 and 0.4, we can surmise that the optimal value is around 0.2.

## V. CONCLUSION

This paper highlights the transformative potential of large language models in generating, processing, and retrieving information while applying a framework to mitigate misinformation. Our proposed Elephant model is fine-tuned from two popular LLMs, GPT3.5 and LLaMA2. In addition, using topic and keyword extraction alongside a cosine similarity metric, our proposed model can mitigate the challenge of misinformation in our digital society via accurate recantations of information.

We collected podcast datasets, processed the data, and fine-tuned two models following our approach. Then, two experiments were conducted to verify the effectiveness of our method. In the first experiment done using the LLaMA 2 variant, after extracting semantic meaning from the prompt and comparing it with the database, the model's output saw an average similarity of 88.29% (0.76588) in comparison to the reference text at a temperature of 0.5. In the second experiment done using the GPT 3.5 variant, after the prompt deconstruction and similarity selection phases, the model's output saw an average similarity of 90.42% (0.8083) in comparison to the reference text at a temperature of 0.1. There are, however, limitations within this paper. Including comparative analysis, particularly against existing IR methods, would be beneficial. Although that would be optimal, this implementation does apply human verification analysis.

It is no secret that information is power. While the function of compiling information in aggregate to reach a conclusion is innate in all individuals, the ability or availability to do the necessary research may not be a resource readily available. Before making a decision, it is paramount to understand the nature of the decision and the stances of all major parties involved. The proposition of this paper is an attempt to solve this problem.

For future works, it is important to recognize that further research and iterations upon this implementation would result in better performance, highlighted below.

**Data Decisions** It may be beneficial to convert the data into SQL schemas. This would likely improve the relevancy of the data entries initially pulled, allowing for similarity calculations on fewer entries. It is also important to recognize that using more varied data. In the future, other forms of media may be incorporated, as well as other metrics such as perplexity or BLEU score.

**Prompt Deconstruction Analysis** Before extracting the prompt, running an analysis to determine the nature of the query would be useful. If the user requests a previous query after extracting the topics and keywords from the prompt, it may be worth keeping a record of past entries pulled if

that is what the prompt requests. If the user requests more information on a topic, finding it and getting more context may be a separate system to defer to.

## REFERENCES

- [1] V. Bush, "As we may think," *The Atlantic*, 1945.
- [2] L. E. Page and S. M. Brin, "The anatomy of a large-scale hypertextual web search engine," *Computer Networks and ISDN Systems*, 1998.
- [3] G. Salton, *The SMART retrieval system—experiments in automatic document processing*. Prentice-Hall, Inc., 1971.
- [4] K. S. Jones, "The role of artificial intelligence in information retrieval," *Journal of the American Society for Information Science*, vol. 42, no. 8, pp. 558–565, 1991.
- [5] —, "What is the role of nlp in text retrieval?" in *Natural language information retrieval*. Springer, 1999, pp. 1–24.
- [6] —, "Information retrieval and digital libraries: lessons of research," in *Proceedings of the 2006 international workshop on Research issues in digital libraries*, 2006, pp. 1–7.
- [7] M. Lease, "Natural language processing for information retrieval: the time is ripe (again)," in *Proceedings of the ACM first Ph. D. workshop in CIKM*, 2007, pp. 1–8.
- [8] M. Unterkalmsteiner, T. Gorschek, R. Feldt, and N. Lavesson, "Large-scale information retrieval in software engineering—an experience report from industrial application," *Empirical Software Engineering*, vol. 21, pp. 2324–2365, 2016.
- [9] S. Jiang, S. Shen, M. Agrawal, B. Lam, N. Kurtzman, S. Hornig, D. Karger, and D. Sontag, "Conceptualizing machine learning for dynamic information retrieval of electronic health record notes," *arXiv preprint arXiv:2308.08494*, 2023.
- [10] Facebook, "Facebookresearch/llama: Inference code for llama models," <https://github.com/facebookresearch/llama>, 2023, accessed: 21-11-2023.
- [11] OpenAI, "Openai," <https://openai.com/>, 2023, accessed: 21-11-2023.
- [12] Y. Zhu, H. Yuan, S. Wang, J. Liu, W. Liu, C. Deng, Z. Dou, and J.-R. Wen, "Large language models for information retrieval: A survey," *arXiv preprint arXiv:2308.07107*, 2023.
- [13] R. Peng, K. Liu, P. Yang, Z. Yuan, and S. Li, "Embedding-based retrieval with llm for effective agriculture information extracting from unstructured data," *arXiv preprint arXiv:2308.03107*, 2023.
- [14] Y. Gong and G. Cosma, "Boon: A neural search engine for cross-modal information retrieval," *arXiv preprint arXiv:2307.14240*, 2023.
- [15] Q. Ai, T. Bai, Z. Cao, Y. Chang, J. Chen, Z. Chen, Z. Cheng, S. Dong, Z. Dou, F. Feng *et al.*, "Information retrieval meets large language models: A strategic report from chinese ir community," *AI Open*, vol. 4, pp. 80–90, 2023.
- [16] H. Abonizio, L. Bonifacio, V. Jeronymo, R. Lotufo, J. Zavrel, and R. Nogueira, "Inpars toolkit: A unified and reproducible synthetic data generation pipeline for neural information retrieval," *arXiv preprint arXiv:2307.04601*, 2023.
- [17] V. Jeronymo, L. Bonifacio, H. Abonizio, M. Fadaee, R. Lotufo, J. Zavrel, and R. Nogueira, "Inpars-v2: Large language models as efficient dataset generators for information retrieval," *arXiv preprint arXiv:2301.01820*, 2023.
- [18] D. S. Chamath Palihapitiya, Jason Calacanis and D. Friedberg, "All-in with Chamath, Jason, sacks & Sriedberg," <https://www.allinpodcast.co/>, [Accessed 21-11-2023].
- [19] spaCy, "Spacy · industrial-strength natural language processing in python," <https://spacy.io/>, 2023, accessed: 21-11-2023.
- [20] R. Rehũfek and P. Sojka, "Software Framework for Topic Modelling with Large Corpora," in *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Valletta, Malta: ELRA, May 2010, pp. 45–50, <http://is.muni.cz/publication/884893/en>.
- [21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.
- [22] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, and P. Fung, "Survey of hallucination in natural language generation," *ACM Computing Surveys*, vol. 55, no. 12, pp. 1–38, 2023.
- [23] OpenAI, "Openai platform," <https://platform.openai.com/docs/models>, 2023, accessed: 21-11-2023.