Can Artificial Intelligence Solve the 1947 Flying Saucer Mystery?

By James Carrion, September 25, 2024

If you have read my freely downloadable book **The Roswell Deception** (https://www.keepandshare.com/doc13/21520/the-roswell-deception-by-james-carrion-pdf-7-8-meg?nav=4), you know that one of the theories I propose was that the Flying Saucer summer of 1947 could have been part of a US-led strategic deception operation against the Soviet Union at the inception of the Cold War. One of the strategic goals for this deception could have been to break the Soviet Diplomatic code, just like England broke the German enigma code during WW2.

I proposed that flying saucer stories as well as other newspaper articles of Soviet intelligence interest, were purposely planted in US newspapers as a sophisticated Chosen Plain Text cryptologic attack to break the Soviet Diplomatic Code. To be clear, I do not suggest that newspaper editors across the United States were coerced into this scheme as some grand conspiracy. I show in my book that only the heads of the press associations had to be in on this deception – the Associated Press (AP), the United Press (UP) and the International News Service (INS). The 1947 articles I have transcribed and used in this study are limited to those distributed to newspapers across the United Sates by these press associations.

To understand how this would work, you first must understand the problem that American cryptologists were faced with in 1947. The Russians used a cryptographic code to encipher their communications sent from their US facilities back to Moscow Center. Since the Russians had no direct lines of communication - no radio, satellite or phone connectivity to Moscow - the Russians used the American telegraph companies to send their enciphered messages. What they may not have known was that every enciphered message was being copied and collected by the American code breakers under Project Shamrock https://irp.fas.org/agency/army/mipb/2012_04-owen.pdf.

The Soviets probably suspected their messages were being copied but also thought their messages were unbreakable. The Russians used a one-time-pad system that theoretically was unbreakable if you followed all the rules. But the Russians got sloppy, and they duplicated their one-time pads (a major no-no) and distributed these pads to their various overseas representatives. There were Soviet trade organizations in the US that used the duplicated one-time pads as well as Soviet intel agents from the KGB and the GRU in the US. The true Soviet diplomats who had consulate work to perform also used these duplicated pads.

The Americans and the British soon learned of this duplicate pad distribution and began to exploit it. The secret code breaking effort went through many name changes but eventually settled on the code name Venona https://en.wikipedia.org/wiki/Venona_project .

If you want to understand the exact process the Soviets used to encipher their messages, please read this article https://www.smithsonianmag.com/history/how-cipher-like-soviet-180970032 which explains the process in detail. In this paper, I am going to offer the simplified version that omits some of the steps that are not relevant to understanding how Venona exploited the duplicate one-time pad use.

Let's say you are a KGB agent in New York and need to send a message to Moscow. Let's make the message simple: **Acorn has infiltrated the Manhattan Project**. First lesson. The Soviets didn't use the real names of their own agents or sources in their communications but instead the cover names of individuals and even locations.

Let's say **Acorn** is the cover name for some new source recruited as a spy by rookie KGB agent **Vladimir Danovich**. So, Vladimir would hand off this six word message (written in Russian of course) to his cipher clerk. The clerk would then consult his handy little code book which contained a five-digit number corresponding to each Russian word and a separate spell table for words that had to be spelled out letter by letter. Let's say the message was simplified to 3 words. **Acorn infiltrated Enormous** (Enormous was the Russian cover name for the Manhattan Project). Let's line up each word with its specific five-digit code from the code book.

Acorn	Infiltrated	Enormous
97345	53087	79423

Next, the cipher clerk would grab the next unused page from his one-time pad which consists of QTY 60 random five-digit code groups. Separately on each one-time pad page was a five-digit group that acted as the page number to identify exactly which page in the one-time pad was being used to encipher the message. Let's say that page number was **64382.**

This page number is added at the beginning of the message so now we have:

	Acorn	Infiltrated	Enormous
64382	97345	53087	79423

Then the clerk takes the first 3 five-digit groups from the one-time pad and lines them under the encoded message. This stream of digits is known as the "key." The clerk would then add the encoded message digits to the one-time-pad digits without carry, resulting in the following enciphered message.

						Acorn					**Infiltrated**					**Enormous**				
						97345					**53087**					**79423**				
New York Encoded Message						9	7	3	4	5	5	3	0	8	7	7	9	4	2	3
Onetime Pad Key	6	4	3	8	2	0	7	4	2	5	6	3	1	2	9	8	7	4	3	9
New York Enciphered Message	**6**	**4**	**3**	**8**	**2**	**9**	**4**	**7**	**6**	**0**	**1**	**6**	**1**	**0**	**6**	**5**	**6**	**8**	**5**	**2**

It was this enciphered message that would be sent over the telegraph back to Moscow. Even if the American Code Breakers had intercepted the enciphered message, they could never break the cipher if the one-time pads were truly used just once.

Note that the one-time-pad page number was not enciphered. This allowed the cipher clerk on the receiving end in Moscow to quickly determine which one-time-pad page should be used to decipher the message.

Here's where the plot thickens. Another Russian KGB agent **Mikhail Kandinsky** in San Francisco wants to send his own enciphered message back to Moscow. Unbeknownst to any of these parties, the San Francisco cipher clerk was given an exact copy of the same one-time-pad that was given to the New York cipher clerk. This was a major screw-up on behalf of the organization managing Soviet cryptography.

If Mikhail's message is **San Francisco Source Recruited**, his cipher clerk may simplify this to: **Babylon Source Recruited** (Babylon is the Soviet cover name for San Francisco).

So, the San Francisco cipher clerk consults the code book and finds the code groups that correspond to each word.

Babylon	**Source**	**Recruited**
85213	**29645**	**24362**

Let's say the San Francisco cipher clerk takes page **64382** from his one-time-pad to encipher the message. Since this is a duplicate of the one-time-pad being used in New York, all of the one-time pad key groups are exactly the same. The resulting encipherment is shown below.

	One-time pad page					Babylon						Source						Recruited					
	64382					85213						29645						24362					
San Francisco Encoded Message							8	5	2	1	3		2	9	6	4	5		2	4	3	6	2
Onetime Pad Key	6	4	3	8	2		0	7	4	2	5		6	3	1	2	9		8	7	4	3	9
San Francisco Enciphered Message	6	4	3	8	2		8	2	6	3	8		8	2	7	6	4		0	1	7	9	1

After the American telegraph companies gave a copy of the KGB enciphered messages sent in New York and San Francisco to the American codebreakers, the codebreakers would notice that the NY and San Francisco messages were "in-depth" meaning they were enciphered using a duplicated one-time pad. They could tell that because both messages started with 64382, indicating they were enciphered from the same one-time-pad page.

Let's review what we have learned so far of the Soviet cryptographic system, which consisted of 3 layers.

1. Cover Name Layer - the first layer was the use of cover names. Many individuals working on behalf of Soviet Intelligence were given cover names. Here are some notable historical figures and their Soviet cover names.
    a. **Julius Rosenberg** (atomic spy) – cover name was **LIBERAL** or **ANTENNA**
    b. **Klaus Fuchs** (atomic scientist and spy) – cover name was **REST** or **CHARLES**
2. Encoding Layer – taking the words and cover names to include in the message and assigning each their corresponding five-digit number from a code book. Think of the code book as a database of fixed code groups mapped to words. Example: 00001 = artillery, 00002 = plan, 00003 = secret, etc. If a word had to be spelled out letter by letter, for example Rosenberg's cover name LIBERAL, there was a spell table in the code book that had a different code group for each Latin alphabet letter.
3. Enciphering Layer – the encoded message was then enciphered using a one-time pad.

Imagine you are an American cryptologist in 1947, and the telegraph messages are piling up on your desk. Where do you begin? By peeling away each layer in reverse to get to the next underlying layer.

1. Strip the Encipherment Layer - This could easily be done with messages in-depth (duplicated one-time pads) at least for their common length. If message A was 12 code groups long and was in depth with message B which was 23 code groups long, cryptologists could strip away the key for the first 12 code groups for each message.
2. Strip the Encoding Layer - this is the difficult part as for each code group in-depth, there are 100,000 possible code group pairs. If the cryptologists could successfully split out these pairs to their original values, each 5-digit group would correspond to a word in the Russian code book (or letter from the spell table). But in 1947, according to the NSA, it was a code book the cryptologists did not have and would have to painstakingly recreate out of thin air, word by word.
3. Even if by some incredibly good luck, the cryptologists had a copy of the Soviet code book (perhaps a stolen copy), and the underling plain text message were revealed, any real Soviet identities (spies) were concealed by the last layer, cover names. Knowing a cover name doesn't clue you in on who the real person is.

Let's put ourselves in an American cryptologist's shoes and do the hard work. Again, this has been simplified somewhat to make it easier to understand.

Strip the Encipherment Layer

Here are the messages from New York and San Francisco, enciphered with the same one-time pad.

New York Encoded Message							9	7	3	4	5		5	3	0	8	7		7	9	4	2	3
Onetime Pad Key	6	4	3	8	2		0	7	4	2	5		6	3	1	2	9		8	7	4	3	9
New York Enciphered Message	6	4	3	8	2		9	4	7	6	0		1	6	1	0	6		5	6	8	5	2

San Francisco Encoded Message							8	5	2	1	3		2	9	6	4	5		2	4	3	6	2
Onetime Pad Key	6	4	3	8	2		0	7	4	2	5		6	3	1	2	9		8	7	4	3	9
San Francisco Enciphered Message	6	4	3	8	2		8	2	6	3	8		8	2	7	6	4		0	1	7	9	1

We would start by ignoring the first code group which is the one-time pad page indicator and focus on the subsequent code groups. We line up the enciphered messages and then subtract without borrowing (remember they were enciphered with add without carry) one from the other, digit by digit. That would look like this.

New York Enciphered Message	6	4	3	8	2		9	4	7	6	0		1	6	1	0	6		5	6	8	5	2
San Francisco Enciphered Message	6	4	3	8	2		8	2	6	3	8		8	2	7	6	4		0	1	7	9	1
Subtract Without Borrowing							1	2	1	3	2		3	4	4	4	2		5	5	1	6	1

What we have effectively done is strip away the encipherment layer. The result from the subtraction also represents the difference between the original encoded messages. In other words, if we performed the same subtract without borrowing, but this time subtracting the original New York encoded message and the San Francisco encoded message, the result should be the same as that of subtracting the enciphered messages. Let's prove that.

New York Encoded Message	6	4	3	8	2		9	7	3	4	5		5	3	0	8	7		7	9	4	2	3
San Francisco Encoded Message	6	4	3	8	2		8	5	2	1	3		2	9	6	4	5		2	4	3	6	2
Subtract without borrowing.							1	2	1	3	2		3	4	4	4	2		5	5	1	6	1

Let me illustrate this a bit differently in case you are not following. Subtracting without borrowing the two green enciphered messages would produce the same result as subtracting without borrowing the two blue encoded messages.

New York Encoded Message	6	4	3	8	2		9	7	3	4	5		5	3	0	8	7		7	9	4	2	3
Onetime Pad Key							0	7	4	2	5		6	3	1	2	9		8	7	4	3	9
New York Enciphered Message	6	4	3	8	2		9	4	7	6	0		1	6	1	0	6		5	6	8	5	2

San Francisco Encoded Message	6	4	3	8	2		8	5	2	1	3		2	9	6	4	5		2	4	3	6	2
Onetime Pad Key							0	7	4	2	5		6	3	1	2	9		8	7	4	3	9
San Francisco Enciphered Message	6	4	3	8	2		8	2	6	3	8		8	2	7	6	4		0	1	7	9	1

Note: Subtracting without borrowing the green enciphered messages from each other or the blue encoded messages from each other produces the same result.

Differencing result. See note above.							1	2	1	3	2		3	4	4	4	2		5	5	1	6	1

Cryptologists examining two messages in-depth had the green enciphered messages in front of them and could easily perform the subtraction operation to get the results. But they had no easy way to take those differencing results and derive what the blue encoded messages were.

The cryptologists could essentially strip away the green encipherment by using this process but are left with just the differencing results 12132 34442 55151. They don't know for each differencing result what the corresponding blue pairs are that make up the original encoded message, as there are 100,000 pair possibilities for each group.

In other words, for the first differencing result 12132, the number of possible pairs of 5-digit numbers that could be subtracted without borrowing from each other and result in 12132 is 100,000 pairs, with 97345 and 85213 just one possible pair of these 100,000. If all we know is the difference of 12132, we have 100,000 guesses at what the original encoded group pair is. Yikes!!!

But what if I already knew one of the encoded message groups? Let's say I knew that from the New York message, the first encoded group was 97345. Well then figuring out what the unknown encoded group from the San Francisco message would be easy as we take our known number 97345 and subtract without borrowing our difference value of 12132 and that reveals 85213, the corresponding encoded group from the San Francisco message.

Figuring Out the Original Encoded Messages

To recap; after the encipherment is stripped away and reveals the differencing results, we then try to determine the encoded message group pairs for each differencing result, where each pair is one of 100,000 possibilities.

Rather than trying to brute force guess which of the 100,000 possible pairs is the right one, could we speed things up?

One method we could use is what is known in cryptology as a Chosen Plaintext Attack. We plant a newspaper story that we believe a Soviet agent may be interested in sending back to Moscow Center as intelligence.

For example, let's say one of our two fictitious KGB agents, Vladimir Danovich from New York sees the following article in the New York Times:

> IRAN HAS ITS DISC PROBLEM
>
> TEHRAN, July 9. AP - The flying saucer fever spread to Iran today.
>
> Press reports from Zabool, Shosef and Sarbisheh near the Afghan frontier said residents there had observed strange "starlike bodies" in the sky which exploded loudly, leaving a cloud of smoke.
>
> The " newspaper Mehri Iran said the objects apparently had something to do with a secret weapon, which it dubbed "V-20."

If Vladimir sends this article to Moscow Center by enciphered telegraph, and the same one-time pad page is also used for another KGB message sent by KGB agent Mikhail Kandinsky in San Francisco to encrypt a different planted news story, American cryptologists now have a very powerful cryptologic inroad.

They could strip the encipherment layer and through trial-and-error figure out the underlying encoded group pairs (each group with 100,000 possibilities) and rebuild the Soviet code book out of thin air. Once they had the code book recreated, it would help them break those messages that were of real intelligence value, not based on planted newspaper articles. One of the trial-and-error methods the cryptologists could use to guess the encoding pair groups is called crib dragging.

Crib Dragging.

Crib dragging is a technique used in cryptanalysis to help break codes or ciphers. A crib is a piece of known or guessed plaintext that corresponds to a section of ciphertext. It's like a small hint or starting point. Dragging refers to moving this known piece of text along the ciphertext to find where it fits best. At each position, you use the crib and the ciphertext to see what the encoded groups could be if your guess is correct. You look for positions that look meaningful or reveals patterns.

I proposed in my book *The Roswell Deception* that in the summer of 1947, planted “flying saucer” newspaper articles was part of a chosen plaintext attack, for the purpose of creating “cribs”. Planted newspaper articles of interest to Soviet Intelligence agents in the US, enciphered and sent back to Moscow via the telegraph companies, could be used as "cribs" by American cryptologists.

Getting the Ciphertext:

Since the Soviets probably believed flying saucers were some new airborne American weapon, they would have been anxious for their agents on American soil to report back flying saucer intelligence. The only secure way they had to do that quickly was sending enciphered telegraphs. Project Shamrock provided these enciphered messages back to the Americans, who now had both the plaintext (the planted newspaper articles) and the cipher texts (the enciphered telegrams).

Crib Dragging Process:

After stripping the encipherment layer off for messages in-depth (duplicated one-time pad use), American cryptologists could "drag" the known plaintext (the crib) along the ciphertext. In essence, crib dragging in a chosen plaintext attack is like using a key you crafted (your chosen plaintext) to probe the lock (the encryption system), helping you understand how the lock works and potentially open other similar locks (decrypt other messages).

Speeding Up Crib Dragging

Crib dragging could be enhanced if you have an anchor point in the enciphered message to start off your analysis with. In the earlier flying saucer article, a word that would be guaranteed to be enciphered by the Soviet agents would be the name of the secret weapon mentioned, the "V-20."

Since the Soviet code book would not have a pre-assigned code group for this name, the cipher clerks would have spelled it out using the spell table of the code book. This was used to spell out anglicized names. Each letter in the Latin alphabet was assigned its own code group and there was a code group for “Start Spell” and “End Spell”. So in this example, “V-20” would have been encoded as follows.

StartSpell	**V**	**2**	**0**	**EndSpell**
02987	34299	09087	11342	83098

Anything encoded with the spell table could act as a starting anchor point for analysis as it constitutes the low-hanging cryptologic fruit. American cryptologists could first attempt to reveal the spell table code groups and then once they isolated them, guess from the known plain texts the words that preceded or followed. Anytime the cryptologists saw the code group 02987, they would know a word was being spelled out, with the end of the word followed by code group 83098.

Spelled-out words then became analogous to the cartouches in Egyptian hieroglyphics which designated the names of Egyptian royals. The more words that the Soviets were forced to use the spell table for, the more anchor points American cryptologists had to guess at the surrounding code groups.

I apologize for this very lengthy lesson in 1947 cryptology, and indeed there is more complexity to it than my simplified version, but it was necessary so you can understand where Artificial Intelligence can step in to solve the 1947 saucer stories.

I told you earlier that the Soviets did not like including the real names of their agents in their encoded messages, preferring to use cover names when referencing their contacts and agents. The exception would be when they included a name that was not an asset or did not have a pre-assigned code name. For example, Venona cryptologists discovered a list of prominent Manhattan Project scientists in some of the Soviet messages that were not covered by a code name, but literally spelled out letter by letter using the spell table.

What if American cryptologists knew of some of these Soviet cover names and purposely planted these proper names in the flying saucer newspaper articles? This would allow cryptologists to match planted messages with real messages where the same cover name was used. For example, if there was a known Soviet agent with cover name John, then planting the first name John in the saucer stories could be of cryptologic benefit.

What would be needed to test this hypothesis would be:

1. A list of known compiled cover names of Soviet agents. Historian and researcher John Earl Haynes compiled exactly that in his *Cover Name, Cryptonym, Pseudonym, and Real Name Index* available on his website at: https://www.johnearlhaynes.org/page66.html a Cover Name, Cryptonym, Pseudonym, and Real Name Index.
2. A corpus of the flying saucer newspaper articles from 1947. Since the height of this activity was from late June to mid-July 1947, I gathered newspaper articles from this time frame (AP, UP, INS press association articles) and then painstakingly transcribed by hand the flying stories to text files.
3. A narrow Artificial Intelligence that could compare the proper names extracted from the 1947 newspaper articles and match them to the cover names from the Haynes's index and then calculate the probability that the matches were statistically significant or just coincidental (random chance).

But the real 1947 newspaper data I had available to me was not as extensive as I would have liked. I decided to supplement the real flying saucer articles with artificial data that I had an AI create. Why? Because, if the flying saucer articles were purposely planted, they must have been optimized for a chosen plaintext attack and I could get AI to do the same.

I started the artificial data generation process by first asking ChatGPT for some advice.

"I want to create a large dataset of artificially generated newspaper articles that are optimized for the purpose of cryptologic gardening. How could I generate this large dataset?"

ChatGPT was kind enough to reply:

Creating a large dataset of artificially generated newspaper articles optimized for cryptologic gardening requires a few key steps. These include text generation, ensuring diverse and realistic content, and encoding the articles in ways that are useful for cryptologic analysis. Here's a comprehensive approach:

1. Text Generation using Language Models

- **Customization:**
    - Generate articles on various topics (politics, sports, science, etc.) to mimic a real newspaper. Ensure that the vocabulary, sentence structure, and content style vary across articles.
    - Include common newspaper elements like headlines, subheadlines, quotes, and bylines.

2. Optimization for Cryptologic Gardening

- **Diverse Text Patterns:**
  - Vary sentence lengths, word choice, and paragraph structure to create diverse text patterns. This helps in creating rich bigram and trigram distributions, which are useful for cryptologic analysis.
- **Incorporate Common Phrases:**
  - Include common phrases, idioms, and clichés found in newspaper writing to reflect realistic text patterns. Also, deliberately include cryptologic phrases or structured sequences that could serve as plaintexts in encryption schemes.

3. Automation and Scaling

- **Automate Article Generation:**
  - Create a script or use a platform to automate the generation of articles. You can set up a loop that generates thousands of articles, ensuring each one is unique and diverse.
- **Dataset Structuring:**
  - Organize the generated dataset into directories based on topics or encoding methods. Maintain a consistent naming convention for easy retrieval and analysis.

4. Quality Control

- **Review Generated Text:**
  - Randomly sample and review generated articles to ensure they meet the desired quality and diversity.

I used a different AI, Groq instead of ChatGPT for the actual artificial data generation. After getting an API key from Groq https://groq.com/ and creating a Python script that would request all the data I needed. I set it loose on Groq by requesting it create 10,000 simulated newspaper articles from 1947 that were optimized for cryptologic gardening (planted newspaper articles) on any topic - not even mentioning “flying saucers” - and then ran a second request for 10,000 simulated newspaper articles from 1947 on just any mundane topic but without any crypto gardening optimization. Each generated newspaper article was saved into its own text file.

This is the Groq query I ran for the 10,000 crypto-gardened optimized newspaper articles:

“You are a 1947 American cryptologist. Your target is Soviet Intelligence. Create a 400 word or less newspaper article to be planted in an American newspaper. This should be a single article using a single subject. Use commonly used phrases, sentence structures and cliches typical of news articles from 1947. Incorporate common and high-frequency words. Embed specific keywords or phrases that are expected to be of particular interest to the encryption target. Article must be plausible. Deliberate use of words with diverse letter frequencies with unique or less common letters. Repetition of key word and phrases. Strategic placement of common digraphs and trigraphs. Inclusion of proper names of varying lengths that are repeated through the article. Use proper names from a variety of linguistic and cultural origins. The article should read exactly as it would in the newspaper. The article should appear to be verbatim from a 1947 newspaper.”

And this is the Groq query I ran for the 10,000 mundane newspaper articles.

"You are a 1947 American Newspaper reporter. Create a 400 word or less newspaper article on random subjects. This should be a single article using a single subject. Use commonly used phrases, sentence structures and cliches typical of news articles from 1947. Article must be plausible. Inclusion of proper names of varying lengths that are repeated through the article. Use proper names from a variety of linguistic and cultural origins. The article should read exactly as it would in the newspaper. The article should appear to be verbatim from a 1947 newspaper."

I set my script to loop and came back hours later and I had 20,000 artificially generated very plausible 1947 newspaper articles to work with.

I placed the real 1947 flying saucer articles extracted from newspapers in a folder labeled 1947SaucerArticles, the Groq-created artificial 10,000 crypto-gardened optimized newspaper articles in a folder labeled GardenedArticles and the Groq-created artificial 10,000 mundane newspaper articles in a folder labeled NotGardenedArticles.

I then wrote a Python script to perform the following steps:

- Randomly select within each folder the newspaper articles to process.
- Extract from each selected article only the proper names of persons and then check them against the Haynes’ index of Soviet cover names to see if it could find matches.
- Run statistical analysis and once the name matches reached statistical significance to a high degree, it would stop processing additional articles in that folder and move

on to the next folder. The Saucers folder was processed first, then the Gardened folder and lastly the Not Gardened folder.

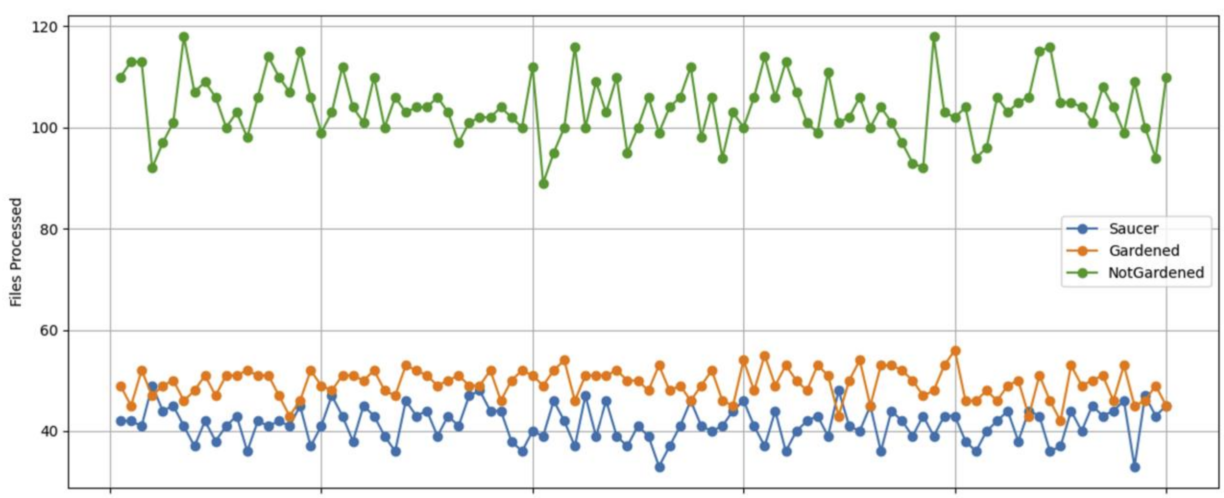
- The script could be set for multiple runs.

I created two different versions of the script, one that applied binomial statistical analysis, and a second script that applied fisher's exact statistical analysis.

Before I reveal the results of my testing, let me preface it by saying that if you feed this model any large enough corpus of text, it will eventually match enough names in the Hayne's cover name index to show a statistical significance. I was interested in knowing how large the corpus of text had to be to reach statistical significance, so I focused my analysis on:

- How many files from each corpus of data (real saucer articles, artificially created gardened articles and artificially created not gardened articles) would have to be processed to achieve statistical significance?
- How would each corpus' result correlate with the other corpuses?

I ran each script for 100 runs and had it output a line chart to show the correlation of results. For the **Binomial Test Analysis, here are the results for 100 runs.** Note the very close correlation between the saucer articles and the gardened articles where was able to achieve statistical significance around 50 articles. Also note that it had to process twice as many not-gardened articles to achieve statistical significance.



I initially expected that the gardened articles which were artificially generated would have been more closely correlated to the not gardened articles, also artificially generated, but that was not the case. Instead, the artificially generated gardened articles more closely correlated with the real saucer articles.

What is causing this odd correlation? How did Groq get the gardened articles to match the Haynes' index to a statistical significance at half the rate of the non-gardened articles, when both corpuses are of equal length and contain equal numbers of proper names to compare?

I am not a statistician, nor am I a data scientist, so I welcome the input of those who are and can analyze and interpret this data further. Perhaps it is in the nature of the Groq model where this mystery lies.

Groq which generated the artificial data is a type of neural network called a transformer. Specifically, it's a variant of the BERT (Bidirectional Encoder Representations from Transformers) model, which is a pre-trained language model developed by Google.

Transformer-based neural networks are generally considered black box models, which means that it is often difficult to understand how it formulate its responses. Here's why:

1. Complexity: Transformer models, such as BERT, GPT, and their variants, have a very complex internal architecture with multiple layers of attention mechanisms, feed-forward networks, and other sophisticated components. This complexity makes it difficult to fully understand and interpret the internal workings of the model.

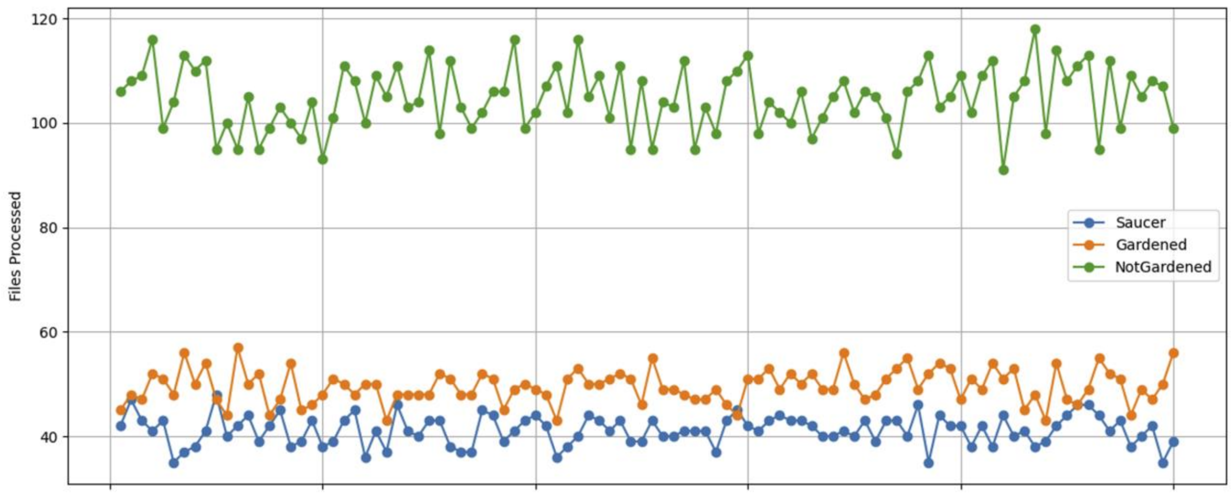
2. Lack of interpretability: Transformer models are typically trained in an end-to-end fashion, where the input data is transformed through multiple nonlinear layers to produce the desired output. This process is not easily interpretable, as it's difficult to trace how the input features are being processed and combined to generate the final output.

3. Opaque decision-making: The decisions made by transformer models are often not easily explainable. The model's outputs are the result of complex interactions between the various components, making it challenging to understand the specific reasons behind a particular prediction or decision.

4. Lack of transparency: Transformer models are often treated as black boxes, where the internal logic and decision-making process are not readily accessible or understandable to the end-user. This lack of transparency can be a concern in applications where the model's decision-making process needs to be explainable, such as in high-stakes decision-making or sensitive domains.

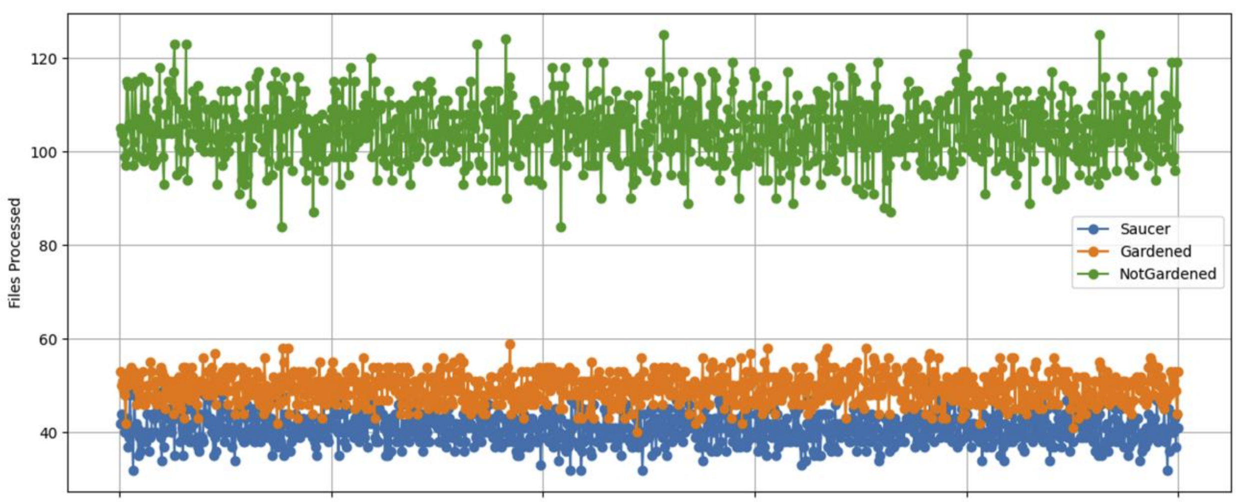
When I asked Groq to put itself in the shoes of a 1947 American Cryptologist targeting Soviet Intelligence in 1947, it is not clear how it generated the resulting artificial newspaper articles. For example, Groq could have found and used the Hayne's index of Soviet cover names and injected those into its articles, even though I didn't ask it to specifically do that. There is simply no way to understand it's complicated reasoning.

For the **Fisher Exact Analysis, the results of 100 runs** were very similar to that of the Binomial Analysis, showing the same close correlation of Saucer to Gardened articles.

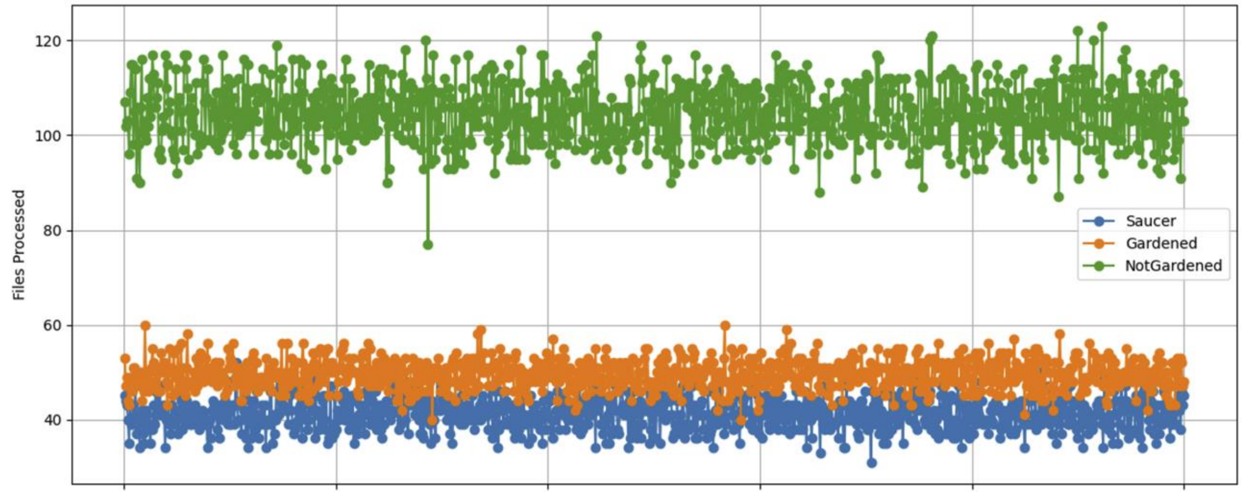


Would anything change if I ran each analysis 1,000 times instead of 100? No, the exact same correlations persist:

For the Binomial Analysis – 1000 Runs



For the Fisher Exact Analysis – 1000 Runs



The correlations are highly intriguing and suggest that there is something here to explore further. Related hypotheses for why flying saucer stories peppered the headlines of US newspapers in 1947 can also be tested by Artificial Intelligence.

For example, I have noticed that proper names in 1947 flying saucer articles would often be misspelled in different newspaper printings from the same newswire articles (Associated Press, United Press, International News Service). In a 1947 flying saucer article that mentions the name Frank Ryman, one newspaper would show the proper spelling of Ryman's last name, while another newspaper would show it spelled as Ryan, even though both newspapers received the same distributed news wire article from the Associated Press.

Was this just a human typographical error or was it purposeful for cryptologic analysis reasons? One way to show if this is statistically significance would be to take a sample of non-flying saucer newswire articles from 1947 and calculate the misspelling error rates for proper names vs the misspelling error rates for flying saucer newswire articles across the newspapers that reprinted the press association distributions.

Artificial Intelligence is changing our world daily and provides a highly useful toolset that can be used for unique applications. This includes historical research and analysis or in the context of this paper, solving one of our oldest mysteries – why in the summer of 1947 was the flying saucer myth born? If you are intrigued as I am by the possibilities, please reach out to me at james_carrion@hotmail.com. If you are a data scientist and wish to review this analysis and crunch the numbers for yourself, I will be happy to provide you a copy of my code and datasets.