GPT as Ally: Improving Cybersecurity Workflows with GPT and Embedding APIs

Kartikeya Sharma

Senior Associate Information Security Engineer @ Equinix

About me

- Currently live and work in Seattle
- Work on the intersection of Data Analytics, AI and Cybersecurity
- Hobbies include Hiking and Drinking overpriced coffee

Disclaimer

The views and opinions expressed in this presentation are my own and do not necessarily reflect the official policy or position of my employer.


The New Risks ChatGPT Poses to Cybersecurity
by Jim Chilton
April 21, 2023


ChatGPT Security Risks: All You Need to Know
ChatGPT security risks, including threats from third-party integrations. Learn best practices for securing implementations and how SentinelOne can assist.


Third-Party ChatGPT Plugins Could Lead to Account Takeovers
Mar 15, 2024
Ravie Lakshmanan
Data Privacy / Artificial Intelligence


Anthropic flags AI's potential to 'automate sophisticated destructive cyber attacks'


Conned by ChatGPT: The Growing Risks of AI-Powered Cyber Attacks

Agenda

1. Foundational Concepts
2. What is ChatGPT and How to use its API?
3. Intelligent Semantic Matching System
4. Application Anomaly Detection System
5. Threat Intelligence Correlation and Analysis System

Foundational Concepts

1. Introduction to Text Embeddings
2. Introduction to Cosine Similarity
3. Introduction to K-Means Clustering
4. Introduction to Prompting

Introduction to Text Embeddings

What are Text Embeddings?

- A way to convert any text into numbers (vectors) that computers can understand
- Can work with different levels of text:
  - Words
  - Sentences
  - Documents
- The numbers preserve the meaning and relationships in the text

Text	Vector Space
"cat"	[0.2, -0.5, ...]
"The cat..."	[0.3, 0.1, ...]
[An Article about Cat]	[0.5, -0.2, ...]

Introduction to Text Embeddings

Word Level

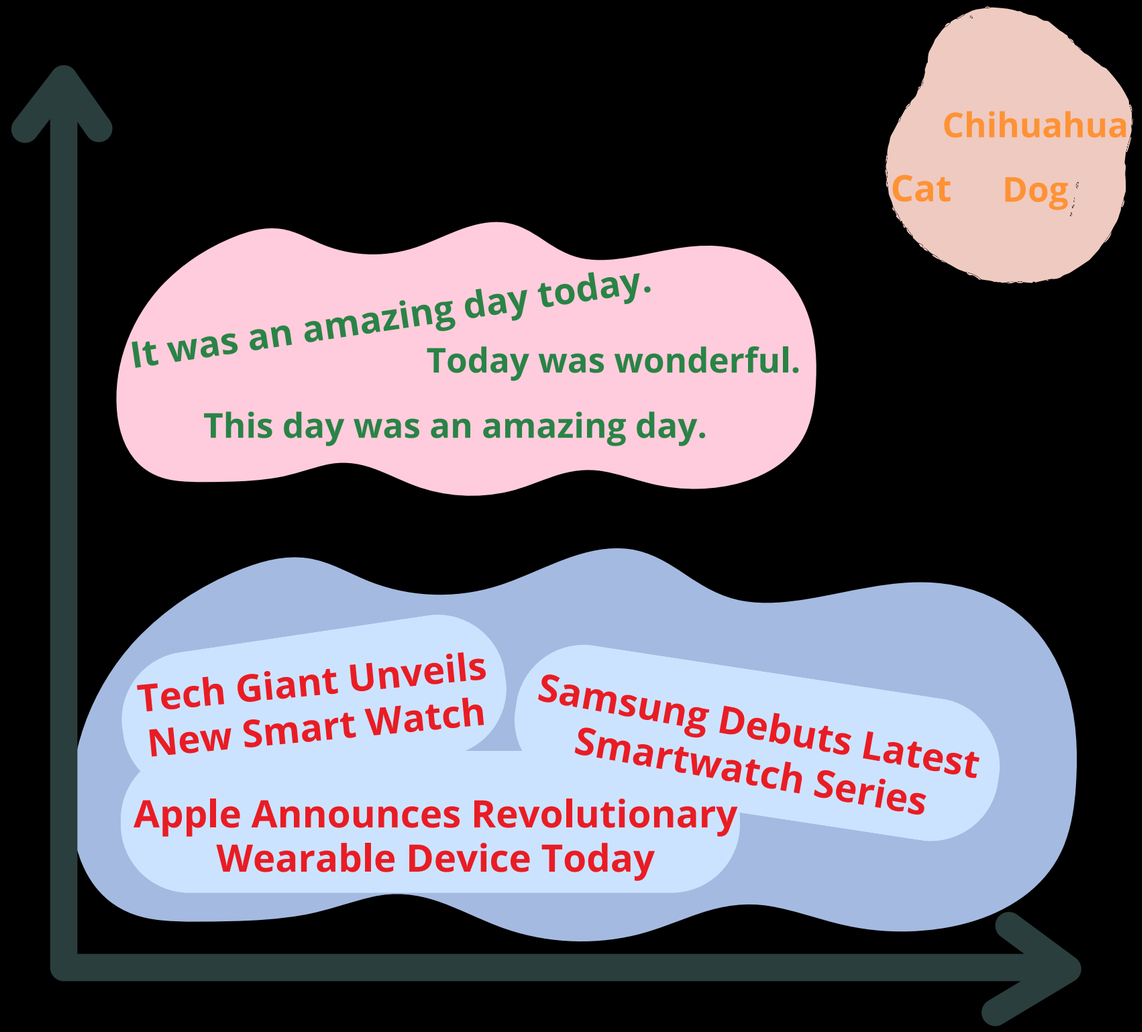
- Individual words get their own vectors
- Similar words get similar vectors

Sentence Level

- Captures the meaning of entire sentences
- Same meaning = similar vectors, even with different words

Document Level

- Can embed entire articles, documents, or books
- Captures overall topics and themes
- Similar documents get similar vectors



Introduction to Text Embeddings

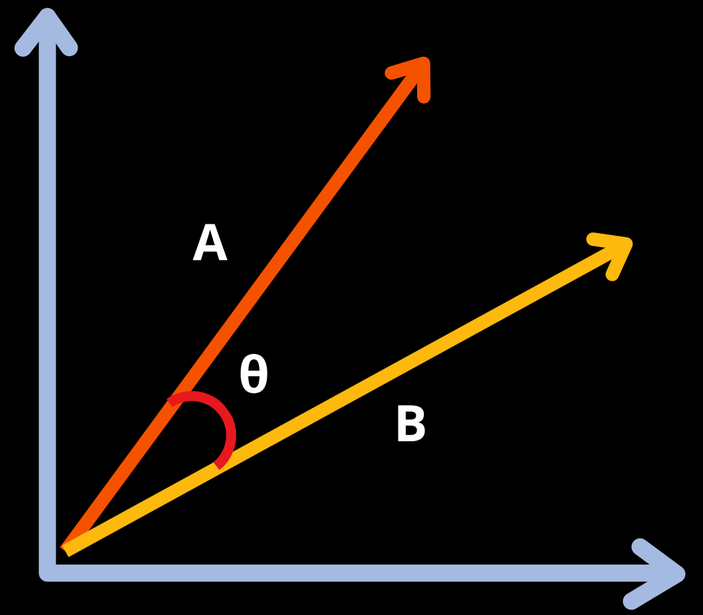
Real-World Applications

- Semantic Search
  - Find documents with similar meaning, not just matching words
- Content Recommendation
  - Group similar articles/documents together
- Document Classification
  - Automatically categorize text by topic
- Question Answering
  - Match questions with relevant answers

Introduction to Cosine Similarity

What is Cosine Similarity?

- A measure of similarity between two vectors
  - Measures the cosine of the angle between vectors
- Output range: -1 to 1
  - 1: Vectors point in same direction (very similar)
  - 0: Vectors are perpendicular (unrelated)
  - -1: Vectors point in opposite directions (opposite)
- Perfect for comparing embeddings!



Introduction to Cosine Similarity

Breaking down the Formula

- A·B: Dot product of the vectors
- ||A||: Length (magnitude) of vector A
- ||B||: Length (magnitude) of vector B
- Normalizes for vector length
- Only considers direction, not magnitude

$$\frac{A \cdot B}{||A|| \, ||B||}$$

Introduction to Cosine Similarity

Advantages for Text Embeddings

- Independent of text length
- Fast to compute
- Works well in high dimensions
- Intuitive interpretation

Text A	Text B	Similarity Score
“Tech Giant Unveils New Watch”	“Apple Announces Revolutionary Wearable Device”	0.89
“Scientists Discover New Planet”	“Astronomers Find Earth-like World”	0.76
“Local Team Wins Championship Game”	“City Sports Club Hosts Tournament”	0.49

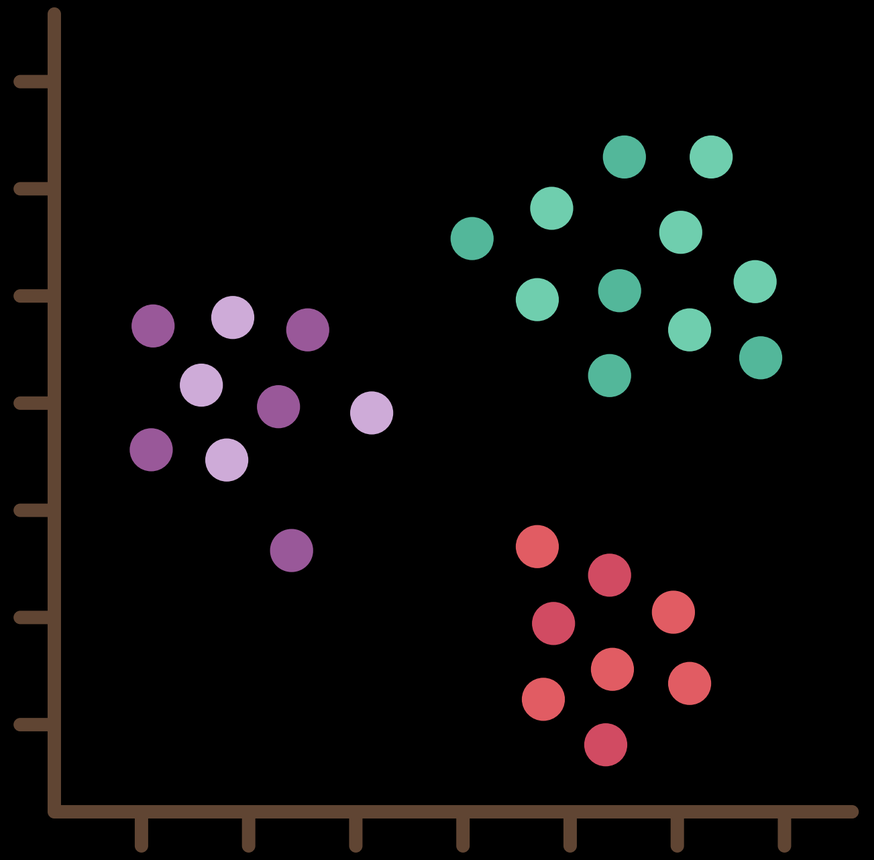
Introduction to K-Means Clustering

What is K-Means Clustering?

- An algorithm to group similar items together
- "K" represents the number of clusters you want
- Perfect for grouping similar text embeddings
- Helps discover natural categories in your data

Key Concepts

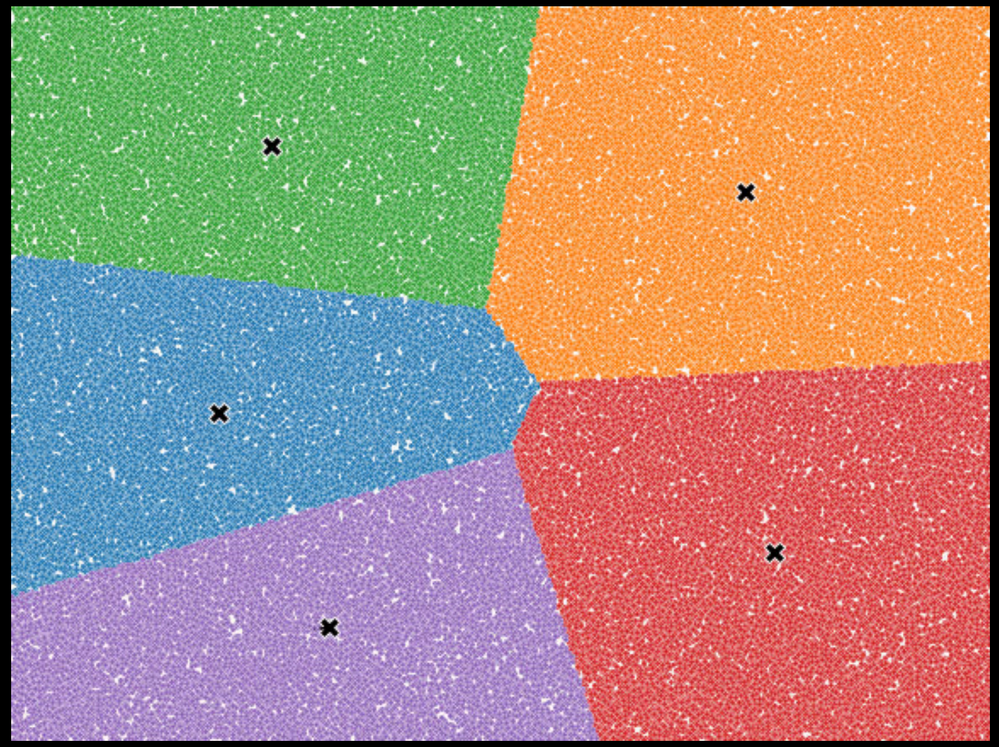
- **Centroids:** The center point of each cluster
- **Distance:** Usually measured using Euclidean distances.
- **Iterations:** Process repeats until clusters stabilize



Introduction to K-Means Clustering

The Algorithm Steps:

1. Choose K (number of clusters)
2. Randomly initialize K centroids
3. Repeat until convergence:
    - Assign each point to nearest centroid
    - Update centroids to center of assigned points
    - Check if centroids moved significantly

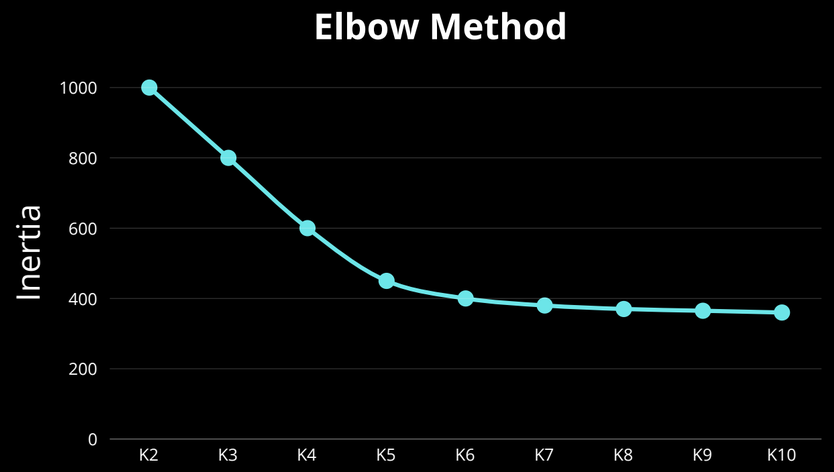


Introduction to K-Means Clustering

Choosing the Right K Value

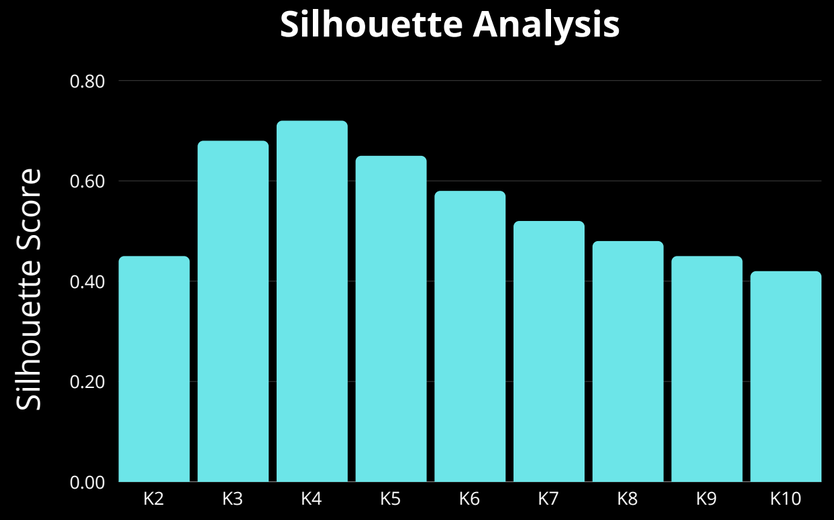
The Elbow Method

- Plot number of clusters (K) vs. Inertia
- Inertia = Sum of squared distances to centroids
- Look for the "elbow" in the curve
- Point where adding more clusters gives diminishing returns



Silhouette Analysis

- Measures how similar points are to their own cluster vs. nearby clusters
- Score range: -1 to 1
  - 1: Well-matched to own cluster
  - 0: On border of two clusters
  - -1: Likely assigned to wrong cluster
- Higher average silhouette score = better K



Introduction to K-Means Clustering

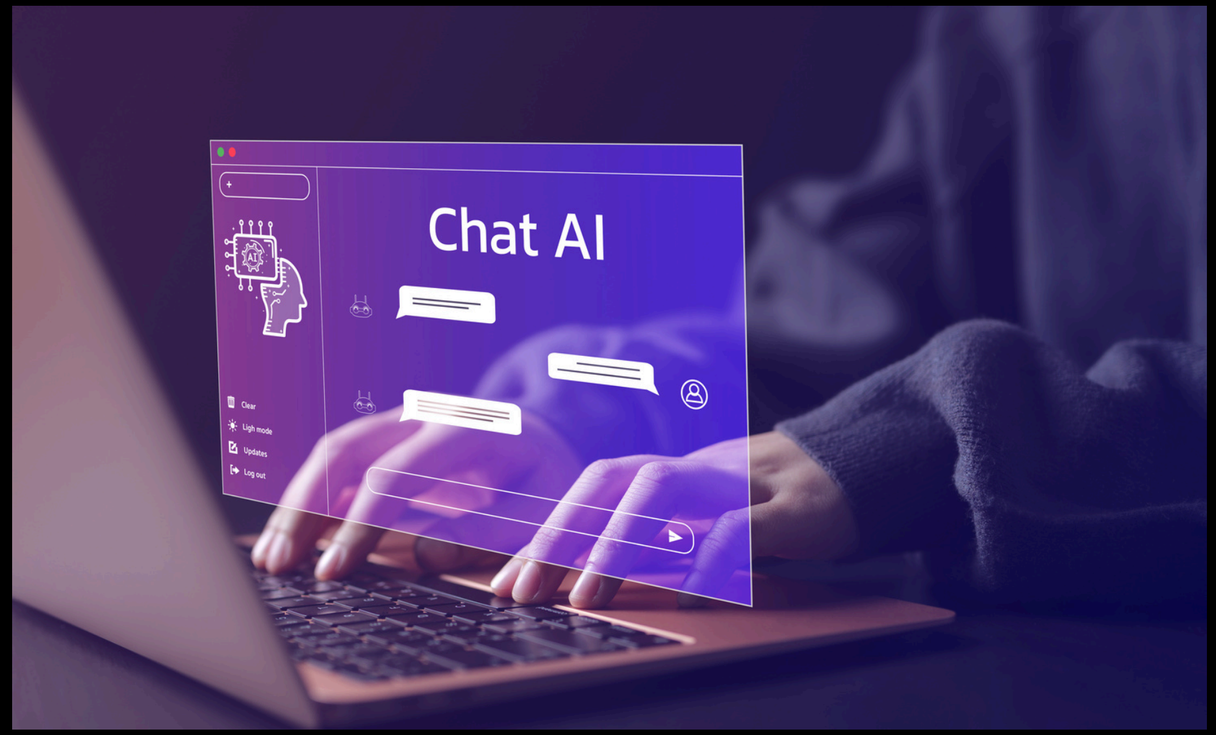
Real-World Applications

- Document Categorization
- News article grouping
- Customer Feedback Analysis
- Content Recommendation
- Topic Discovery

Introduction to Prompting

What Makes a Good Prompt?

- Clear and specific instructions
- Well-structured format
- Consistent style
- Appropriate context
- Purpose-driven design



Introduction to Prompting

Best Practices 
Be Specific
Use Examples
Structure Your Output
Provide Context

What to Avoid 
Not being Specific
Ambiguous Instructions
Inconsistent Format
Missing Context

Introduction to Prompting

Role: You are a market research analyst specializing in consumer technology.

Context: We're analyzing customer reviews for a new smartphone launch. We have a dataset of customer feedback that needs to be categorized.

Task: Analyze the following review and provide:
1. Overall sentiment (positive/negative/neutral)
2. Key themes mentioned (max 3)
3. Most significant feature discussed
4. Any potential product issues highlighted
5. Customer satisfaction score (1-10)

Format your response as JSON with the following structure:

```
{
   "sentiment": string,
   "themes": string[],
   "main_feature": string,
   "issues": string | null,
   "satisfaction_score": number
}
```

Example Output:
Input: "The camera quality is amazing, especially in low light. Battery life could be better though. Overall, very happy with the purchase!"

```
{
   "sentiment": "positive",
   "themes": ["camera quality", "battery life", "overall satisfaction"],
   "main_feature": "camera",
   "issues": "battery life",
   "satisfaction_score": 8
}
```

Now analyze this review:
[Customer review goes here]

What is ChatGPT and How to use its API?

1. What is ChatGPT?
2. Open AI Chat API
3. Open AI Embeddings API

What is ChatGPT?

ChatGPT is a conversational AI developed by OpenAI, designed to understand and generate human-like text.

Capabilities: It can answer questions, engage in natural dialogues, generate human-like text, and assist with a wide range of tasks, making it versatile for various applications.

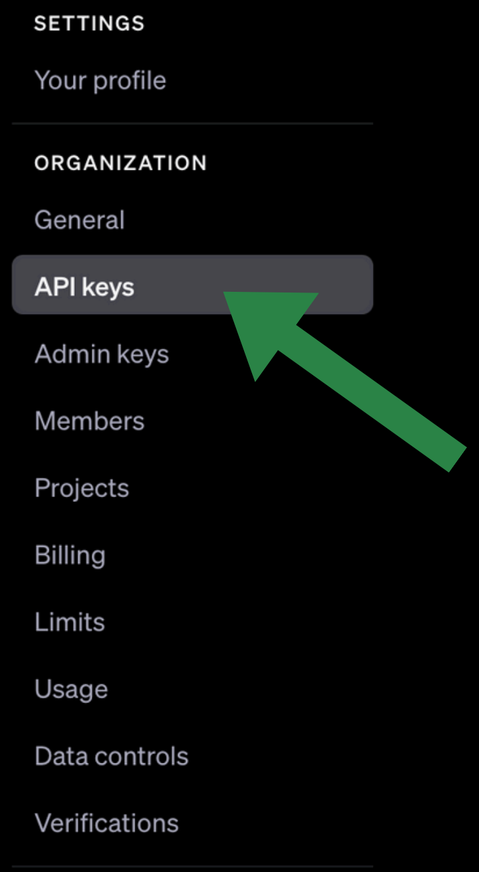
Use Cases: Customer support chatbots, Virtual assistants, Tutoring systems, and Content generation tools.

Overview of Open AI APIs

- **Chat API** Powers conversational AI for interactive dialogues in customer support, virtual assistants, and more.
- **Embeddings API:** Provides vector representations of text for similarity search, recommendations, and clustering tasks.
- **Audio API:** Converts speech to text, enabling transcription and voice-enabled applications.
- **File API:** Allows file upload and management for seamless data integration and fine-tuning.
- **Fine-Tuning API:** Customizes models with specific data to optimize performance for specialized tasks.
- **Moderation API:** Flags harmful or sensitive content, promoting safe and respectful interactions.

Setting Up and Using the API

- Go to https://platform.openai.com/settings/ and create your API Key.
  - Can create Project keys or User Keys
- Choose Your Method for API Calls
  - cURL or Requests (Python)
  - OpenAI Client Libraries:
    - Official client libraries for Python, Node.js and .NET



Overview of Open AI Chat API

It enables developers to integrate ChatGPT's conversational capabilities directly into their own applications, providing a way to create interactive, AI-driven experiences for users.

- **Key Features:** It provides natural language understanding, maintains conversation context, and allows customization through parameters like temperature and max tokens.
- **How It's Used:** Messages are organized by roles (system, user, assistant) to create flexible, multi-turn dialogues tailored to specific use cases.
- **Benefits:** It is scalable, adaptable across various fields, and quick to deploy, enabling fast integration of conversational AI into applications.

Using the Chat API in Python

```python
from openai import OpenAI

client = OpenAI(api_key="YOUR-TOP-SECRET-KEY")

response = client.chat.completions.create(
    model="gpt-4o",
    messages=[
        {"role": "system", "content": "You are a helpful assistant for geography."},
        {"role": "user", "content": "What is the capital of Texas?"}
    ],
    temperature=0.7,
    max_completion_tokens=100
)

print(response.choices[0].message)
```

Important Parameters in Chat API

Parameter	Description
model	Specifies the model to use, e.g., "gpt-4o"
messages	A list of dictionaries that sets the conversation flow with roles (system, user, assistant).
temperature	Controls response randomness; higher values (up to 2) make responses more creative, while lower values make them more focused.
max_completion_tokens	Sets the maximum length of the completion.
top_p	Controls diversity in responses as an alternative to temperature.
n	Defines the number of responses to generate; helpful for getting multiple variations in one call.

Overview of Open AI Embeddings API

The Embeddings API generates high-dimensional vector representations of text, enabling developers to incorporate advanced search, similarity, and clustering functionalities into their applications.

- **Key Features:** It captures the semantic meaning of text as numerical vectors, allowing for tasks like text similarity, document search, and recommendation systems based on language context.
- **How It's Used:** Developers send text input to the API, which returns an embedding vector. This vector can be used to compare text meanings, group similar content, or retrieve relevant information.
- **Benefits:** The Embeddings API is efficient, scalable, and versatile, making it easy to implement robust semantic search and recommendation features across applications and domains.

Using the Embeddings API in Python

```python
from openai import OpenAI
client = OpenAI(api_key="YOUR-TOP-SECRET-KEY")

response = client.embeddings.create(
  model="text-embedding-3-large",
  input="I am in Austin.",
  encoding_format="float",
  dimensions=10
)
print(response.data[0].embedding)
#  [0.099014774, 0.40451068, -0.26185364, 0.3584641, -0.3946019,
#  0.4106308, 0.107102074, 0.32320055, 0.11875944, 0.41849953]

```

Important Parameters in Embedding API

Parameter	Description
model	Specifies the model used for embeddings, e.g., "text-embedding-ada-002"
input	The text or list of texts for which to generate embeddings; can handle multiple inputs for batch processing.
encoding_format	The format to return the embeddings in, e.g., "float" or “base64”

Intelligent Semantic Matching System

1. Overview
2. System Design
3. Example Prompts

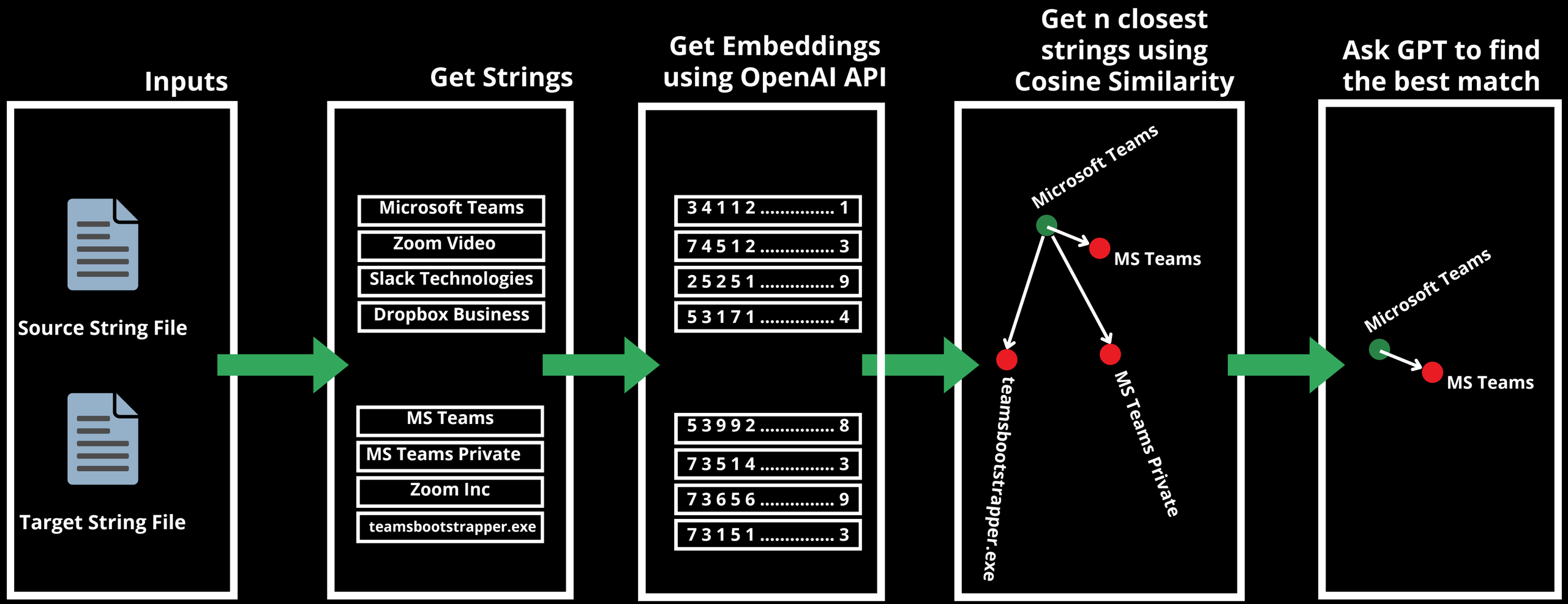
Overview

The Intelligent Semantic Matching System is a pattern recognition tool that leverages Natural Language Processing (NLP) to identify and match semantically similar strings. Through embedding-based comparison and AI-driven validation, it enables accurate matching of text patterns even when they're not exactly identical.

Example Use Cases

- **Domain Name Log Analysis:** Matches domain names found in security logs against allow/deny lists to automate access control decisions and identify potential security threats.
- **Threat Indicator Correlation:** Cross-references and matches threat indicators across different data fields and formats, helping identify related security incidents and patterns.
- **Application Name Matching:** Enables flexible matching of similar but non-identical application name strings, accounting for variations in formatting, spelling, or representation while maintaining accuracy.

System Design



Example prompt 1 (Application Name Matching)

You are an expert system for identifying matching applications despite naming variations. Analyze the source application and potential matches to determine if any represent the same application.

Example Format:
Source: Microsoft Teams (Team collaboration platform)
Potential Matches:

- MS Teams (Enterprise communication tool)
- Microsoft Team (Collaboration software)
- Teams by Microsoft (Video conferencing app)

Expected Output:

- Matching String: MS Teams (Enterprise communication tool)
- Similarity Reason: Same product, slight name variation

Rules:

- Consider common abbreviations and slight misspellings
- Version numbers or editions (e.g., "Pro", "2023") indicate same core product
- Similar products from different companies are not matches
- If no exact match is found, return null
- Keep similarity reasons under 8 words

Return Format:
{"Source": "Microsoft Teams (Team collaboration platform)",
"Matching String": "MS Teams (Enterprise communication tool)",
"Similarity Reason": "Same product, slight name variation"}

Example prompt 2 (Domain Pattern Recognition)

You are an expert at identifying related domain names. Given a source domain, analyze the list of target domains and identify which ones belong to the same organization or service, considering common domain patterns and variations.

Example Format:
Source Domain: akamai.com
Target Domains:

- edgesuite.net
- akamaiedge.net
- cloudfront.net
- fastly.net
- edgekey.net

Expected Output:
Related Domains:

- edgesuite.net (Akamai's content delivery network)
- akamaiedge.net (Edge computing service domain)
- edgekey.net (SSL certificate domain)

Rules:

- Consider service-specific domains that don't match main domain
- Include infrastructure domains (cdn, dns, edge)
- Include technical domains that may look unrelated
- Check for known enterprise service patterns
- Exclude similar service providers
- Provide brief explanation for each match.

Return Format:

```
{   "source": "akamai.com",
    "matches": ["edgesuite.net", "akamaiedge.net", "edgekey.net"],
}
```

Application Anomaly Detection System

1. Overview
2. System Design
3. Walk through the process

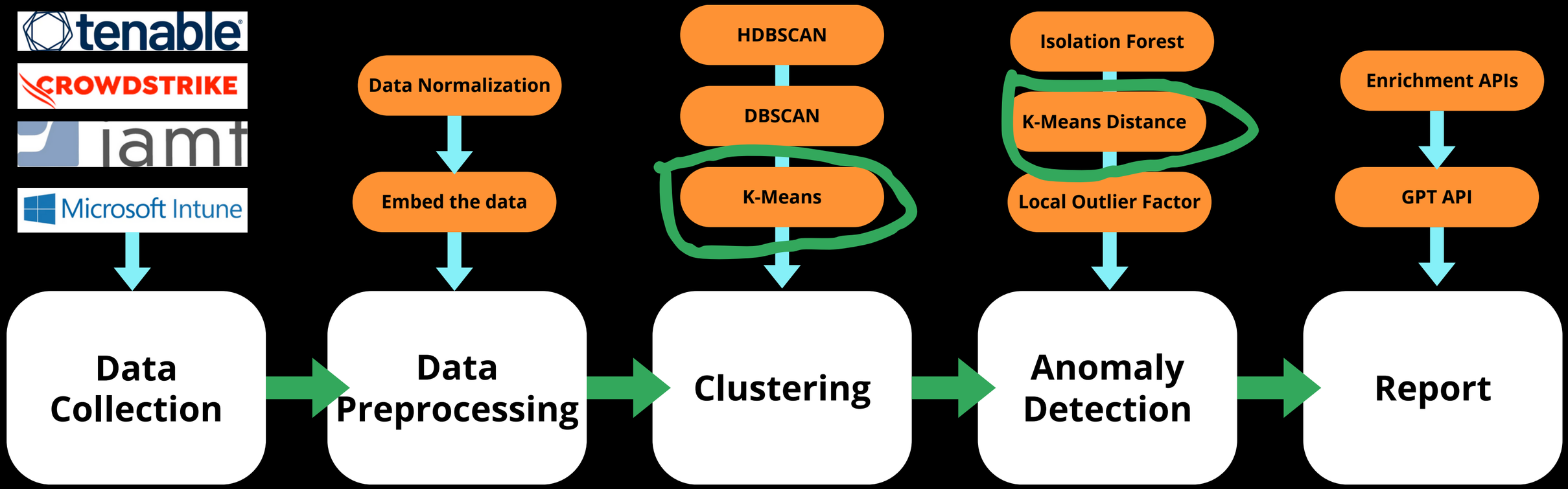
Overview

The Application Anomaly Detection System is an AI-driven tool designed to identify and flag unusual or potentially malicious applications. It uses advanced embedding-based analysis and anomaly detection algorithms to detect deviations from established patterns. By integrating contextual enrichment the system enables detection of applications that pose potential risks.

Example Use Cases

- **Real-Time Anomaly Detection**: Continuously monitors newly detected applications to flag anomalies in real time.
- **Application Risk Scoring**: Assigns a risk score to applications based on clustering and anomaly detection techniques which could help security teams prioritize high-risk applications for further investigation.
- **Allowlist Validation**: Compares flagged applications against an approved list to identify unauthorized software installations.

System Design



Data Collection and Preprocessing

- **Collect** Name, Version and Operating System of an application installed on every Employee’s workstation.
  - Crowdstrike, Tenable, Jamf, Microsoft Intunes etc
  - Cyber Asset Attack Surface Management
- **Preprocessing:**
  - Normalization
    - Application Name Normalization
    - Everything lowercase
  - Aggregation
    - Application count for each employee
  - Calculating/getting embedding for each application
  - Reduce dimensionality using Principal Component Analysis

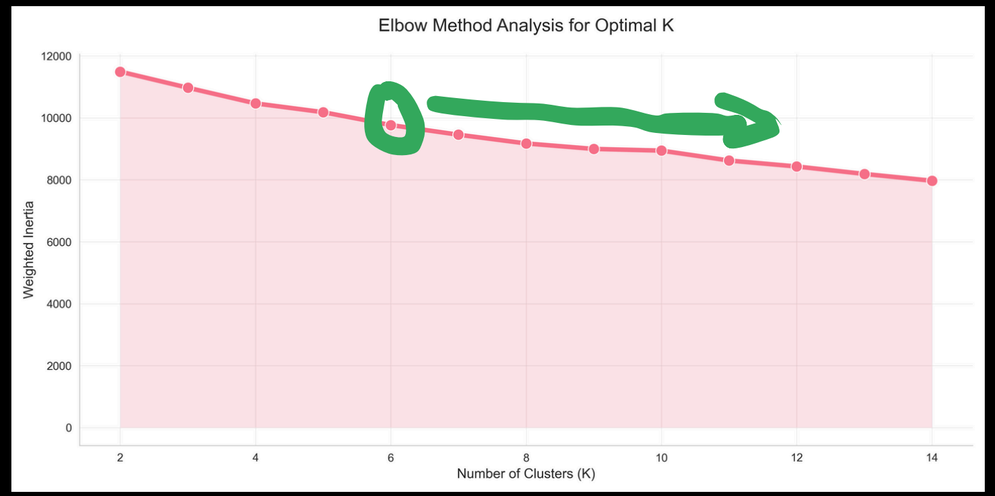
Weighted K-Means

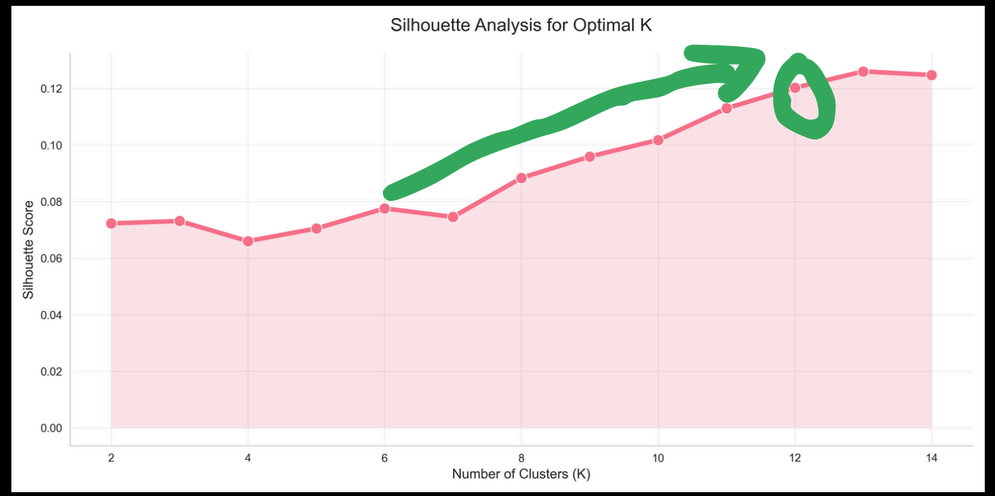
Optimal K Selection:

- Use a random sample of the data to determine the optimal number of clusters (K) to reduce computation time.
- Apply the Elbow method by plotting inertia versus K.
- Alternatively, use the silhouette score to evaluate clustering performance for different K values.

Based on the images on the left, K=12 looks like optimal number of clusters because:

- well past the elbow point, capturing most of the variance reduction
- achieves one of the highest silhouette scores
- Adding more clusters provides minimal improvement in silhouette score

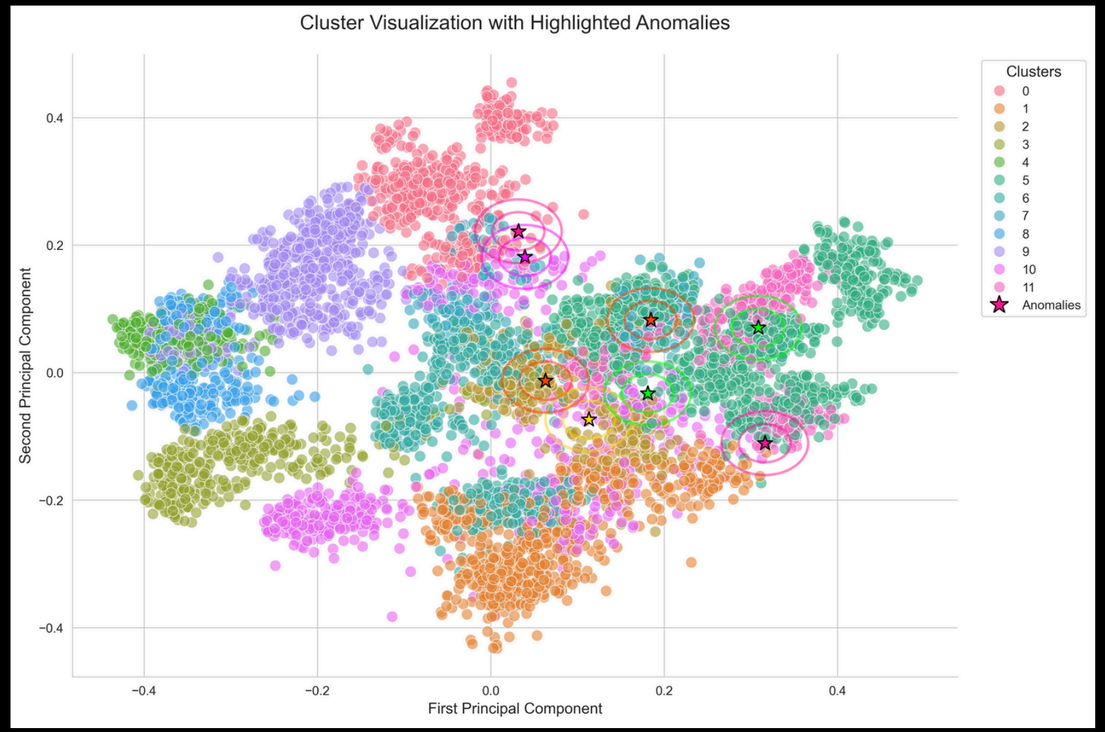




Weighted K-Means

Clustering:

- MiniBatchKMeans if the dataset is huge
- Otherwise normal KMeans
- Use application counts for each workstation as weights.



Anomaly Detection:

- Compute the distance of each data point to its assigned cluster centroid.
- Calculate a threshold for anomalies
  - mean distance plus three standard deviations
- Identify data points (applications) that are farther than the threshold as anomalies.

Reporting

Data Enrichment:

- Use APIs from services like CrowdStrike and Tenable to enrich application data with known vulnerabilities and threat intelligence.
- Cross-check applications against databases such as Palo Alto Applipedia to identify characteristics like "Evasive," "Capable of File Transfer," and other behavioral traits.

Chat API Use Case:

- Threat Reporting: Provide the enriched application data to the Chat API to generate comprehensive threat reports, including potential risks and remediation suggestions.
- Validation Against Allowed Applications: Compare flagged anomalous applications with the approved application list and confirm whether they are authorized.
- Risk Scoring and Prioritization: Ask the Chat API to assign a risk score to anomalous applications and prioritize further investigation based on severity.

What can you do differently?

Explore Alternative Clustering Approaches

- DBSCAN (Density-Based Spatial Clustering of Applications with Noise)
- HDBSCAN (Hierarchical DBSCAN)

Go Beyond Clustering

- Local Outlier Factor (LOF)
- Isolation Forest

Online Detection System

- Develop an online application:
  - Continuously monitor and evaluate new applications in real time.
  - Assign a real-time anomaly score to each new application, determining its risk level dynamically.

Threat Intelligence Correlation and Analysis System

1. Overview
2. System Design
3. Walk through the process

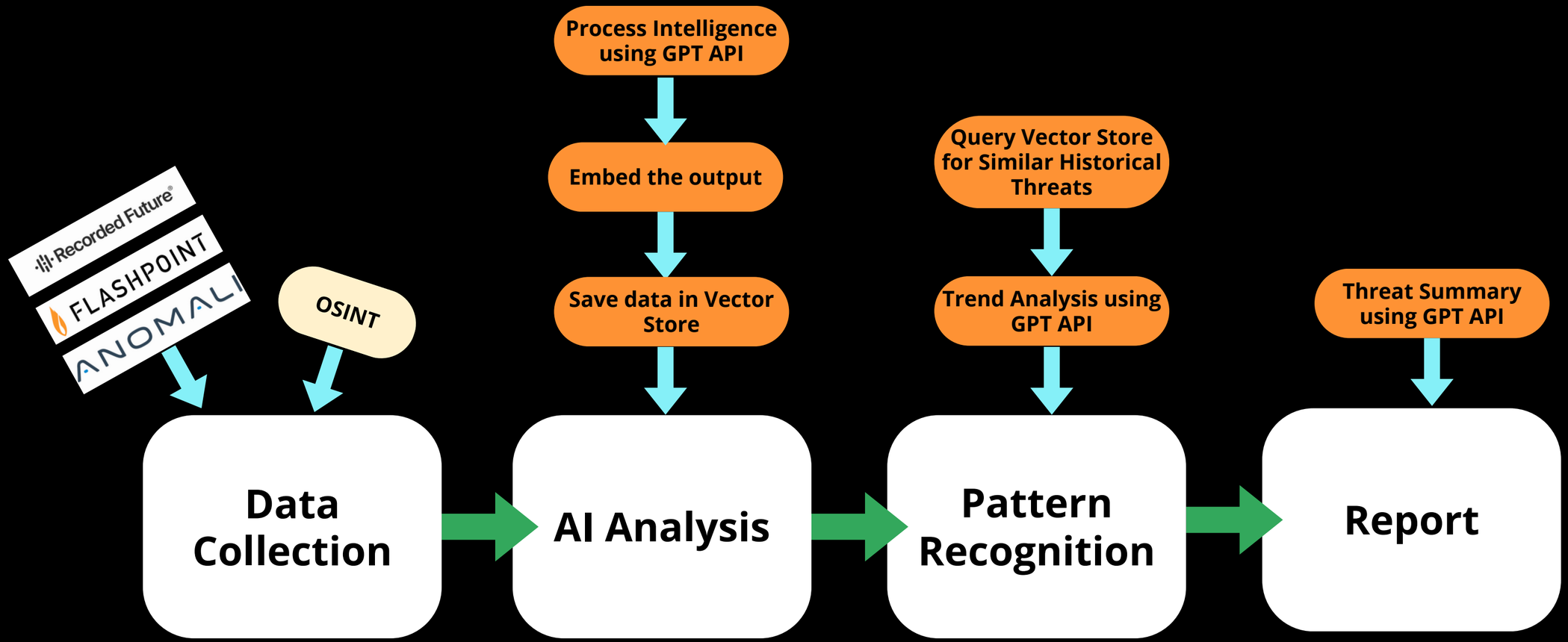
Overview

The Threat Intelligence Correlation and Analysis System is an AI-powered tool designed to detect, analyze, and predict cyber threats targeting organizations. It leverages advanced language models and vector similarity analysis to process threat intelligence from multiple sources and identify emerging attack patterns. By integrating trend analysis with daily threat monitoring, the system enables security teams to stay ahead of potential threats before they materialize into attacks.

Example Use Cases

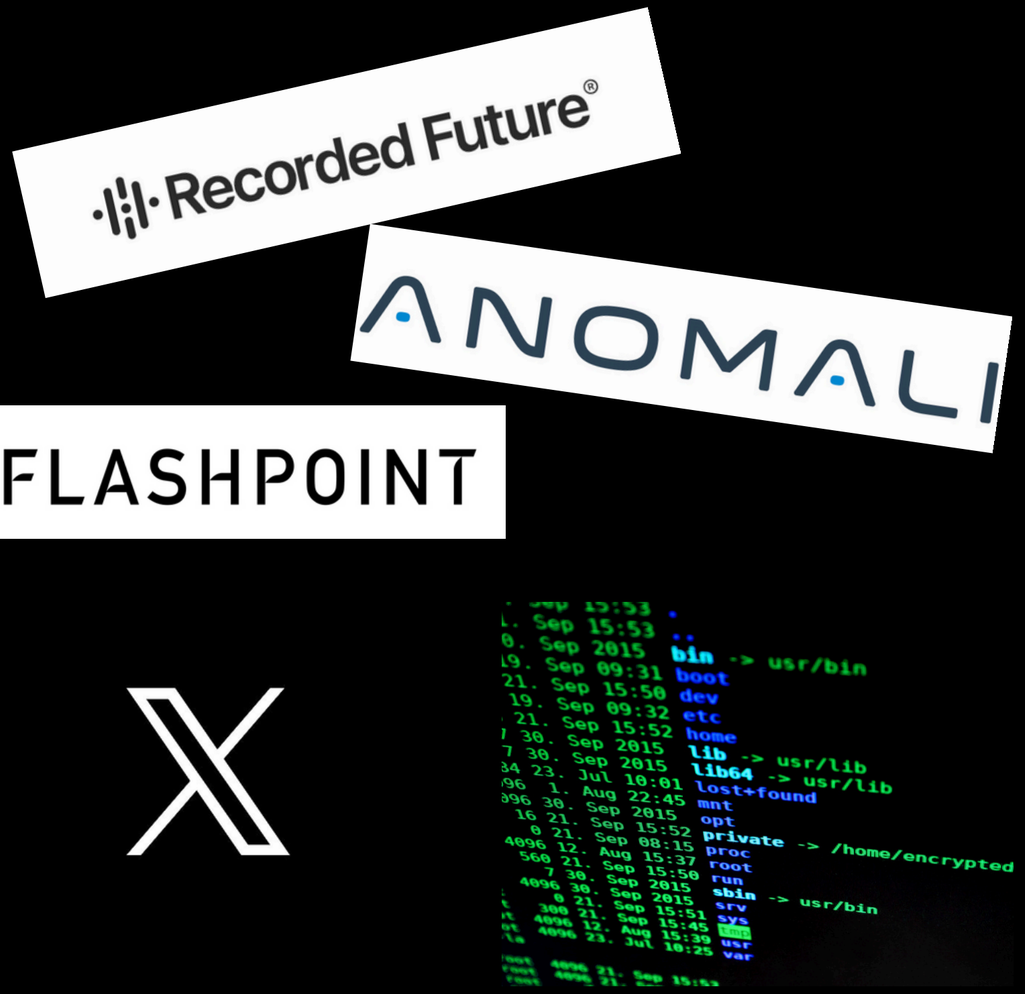
- Early Detection of Supply Chain Attack
- Preventing Financial Fraud Campaign
- Protecting Cloud Infrastructure

System Design



Data Collection

- **Daily collect data from:**
  - **CTI Feeds**
    - Recorded Future
    - Anomali
    - Flashpoint
  - **OSINT**
    - X/Twitter
    - Security Forums
    - Security News Websites



AI Analysis

- **GPT processes all daily intelligence**
  - Generates structured JSON analysis
  - Focuses on company-specific/ industry-specific threats
  - Assigns severity and confidence levels
- **Store each threat to Vector Database Storage**

{ "date": "2024-02-20",
"threats": [
{ "threat_id": "TH-20240220-001",
"threat_type": "Targeted Phishing Campaign",
"description": "Sophisticated phishing campaign targeting consulting firms' project delivery teams. Attackers are using highly customized emails that reference actual client projects and deliverables. Evidence suggests attackers have access to previous consulting project information, possibly from earlier compromises.",
"target_industries": ["Consulting", "Professional Services"],
"severity": "Critical",
"indicators": {
"email_subjects": ["Project Deliverable Review Required", "Urgent: Client Milestone Update", "Project Phase Sign-off Required"],
"attachment_types": ["PDF", "XLSX", "DOCX"],
"malicious_domains": ["secure-project-portal.com", "consulting-deliverables.net"],
"attack_patterns": ["Spear phishing using project terminology", "Valid employee names in sender fields", "References to actual project timelines"] },
"actor": "ProjectSilence",
"confidence": "High",
"sources": ["Recorded Future #2234", "Anomali Alert #785", "OSINT - Multiple researchers"] },}

Pattern Recognition and Reporting

- **Similarity matching with historical threats for each threat**
  - Use Cosine Similarity
- **Trend analysis and prediction for each threat**
- **Daily threat summaries**

As a threat intelligence analyst, review today's threats and their trend analysis to create an actionable daily summary for consulting firm security teams.

Input Data:

1. **Today's Threats: [Insert day's JSON threat data]**
2. **Trend Analysis Results: [Insert JSON from pattern matching and predictions]**

Create a comprehensive daily summary focused on what security teams need to know and do.

Structure the response as:

{ 'date': 'Current date',
'critical_alerts': [{ 'threat': 'Brief description', 'trend_context': 'How this fits identified patterns', 'immediate_actions': ['What teams should do now'] }],
'emerging_threats': [{ 'pattern': 'Identified trend', 'prediction': 'Expected development', 'recommended_preparation': ['Proactive steps'] }],
'daily_focus': ['Prioritized security actions based on both current threats and trends'],
'24hr_outlook': 'Forecast combining current threats and trend predictions' }

Questions?

References

https://www.ibm.com/topics/k-means-clustering

https://www.datastax.com/guides/what-is-cosine-similarity

https://stackoverflow.blog/2023/11/09/an-intuitive-introduction-to-text-embeddings/

https://platform.openai.com/docs/guides/prompt-engineering

https://platform.openai.com/docs/overview

Thank you!