# Technical Report: A2A, MCP, and ML Deployment for Sentiment Analysis Agents

## **1. Introduction**

The growing need to classify user opinions from unstructured text data such as tweets and product reviews has driven the development of automated sentiment analysis systems. This technical report outlines a scalable, cloud-native architecture built on A2A (Agent-to-Agent) communication, a centralized Multi-Agent Control Plane (MCP), and integrated Machine Learning (ML) components. The system supports Twitter and iPhone sentiment analysis via modular agents managed through a shared coordination framework.

## **2. Goals**

* Develop an intelligent, modular system that classifies sentiment across multiple sources.
* Enable scalability and low-latency inference through distributed deployment.
* Empower agents to operate independently while routing intelligently via a central orchestrator.
* Deliver a fully deployable system on Docker/Kubernetes cloud infrastructure.

## **3. Problem Statement**

Organizations face challenges in:

* Extracting sentiment from large volumes of social and review data.
* Maintaining multiple ML services without centralized control.
* Ensuring integration and routing among distributed intelligent agents. This system solves these problems using MCP, agent routing, and automated deployment pipelines.

## **4. Dataset Description**

### **Twitter Dataset**

* Source: Pre-labeled tweets with sentiment classification (Positive, Negative, Neutral).
* Size: ~75,000 samples
* Fields: Tweet ID, Entity, Sentiment, Tweet Content

### **iPhone Review Dataset**

* Source: Amazon product reviews
* Fields: reviewDescription, ratingScore
* Label derivation: Rating 1–2 → Negative, 3 → Neutral, 4–5 → Positive

## **5. Implementation Details**

* **Language**: Python 3.11
* **Libraries**: scikit-learn, pandas, aiohttp, textblob, transformers
* **Model**: Random Forest and BERTweet (Twitter), TextBlob/TF-IDF (iPhone)
* **Vectorization**: TF-IDF for classic models, tokenizer for transformer-based inference

## **6. System Components**

### **MCP Client and Server Integration**

The system is structured such that the MCP server listens for function calls from agent clients. MCP clients act as wrappers for the agents and route communication using either standard input/output (for terminal-based tools) or HTTP. The client uses schema definitions and registered MCP decorators (@mcp.tool) to facilitate execution.

* **MCP Server**: Hosts service registration, manages message routing, handles execution lifecycle.
* **MCP Client**: Sends query via CLI or embedded API (e.g., FastMCP.run("tool\_name", input))
* **Embedding Support**: For routing, sentence transformers or keyword-based vector encoders are used to transform queries into semantic space and match the most appropriate agent.

### **A2A Overview**

A2A (Agent-to-Agent) is a lightweight coordination SDK that enables modular and autonomous agent tools to be invoked via message-based or callable interfaces. Each agent runs as an independent process or module, exposing callable functions through a shared routing framework. Agents do not depend on each other directly, making the architecture fault-tolerant and extensible.

* **Key Features:**
  + Stateless communication over stdio, sockets, or HTTP
  + Lightweight and minimal overhead
  + Integrates seamlessly with MCP for routing and service discovery

### **MCP Overview**

The Multi-Agent Control Plane (MCP) acts as a central orchestrator and message bus for all agents. It facilitates the routing of user requests to appropriate agents, manages agent registration and discovery, and supports rich input/output transformations.

* **Responsibilities:**
  + Handles agent lifecycle and communication
  + Uses decorators like @mcp.tool() to expose functions for coordination
  + Supports interactive CLI, stdio, and programmatic execution via SDK

### **A2A System**

* A lightweight coordination SDK to enable tool-like callable agent interfaces
* Uses FastMCP and message transport (stdio or HTTP)
* a2a\_main.py: Main router script that parses user inputs and calls agents using MCP

### **MCP Server**

* A central server exposing callable tools
* Manages routing logic, agent registration, and async execution
* Developed using FastAPI and mcp.server
* mcp/server/fastmcp.py: Defines FastMCP class and manages tool registration

### **ML Models**

* **Twitter Agent**: Fine-tuned BERTweet Transformer (HuggingFace)
  + a2a\_twitter\_sentiment\_agent.py handles loading the model, tokenizing input, and returning prediction
  + Uses Hugging Face's AutoModelForSequenceClassification
  + Optimized for short-text sentiment classification
  + Example Query: *"Twitter keeps crashing on my phone."*
  + Output: **Negative** sentiment with ~88% classification confidence
* **iPhone Agent**: Random Forest with TF-IDF vectorization and optional TextBlob sentiment analysis
  + a2a\_iphone\_sentiment\_agent.py loads review CSVs, vectorizes reviews, applies RandomForestClassifier
  + Sentiment is derived from Amazon star ratings and review content
  + Alternative lightweight path uses TextBlob polarity scoring
  + Example Query: *"The iPhone battery drains too fast."*
  + Output: **Negative** sentiment (probability from RF classifier or polarity < -0.1 from TextBlob)

### **Routing Agent**

* Uses keyword classification or sentence embeddings to identify target agent
* a2a\_main.py uses mcp.call(tool\_name, input) to route user query
* Routing Accuracy observed: **100%** in test samples

## **7. Workflow**

### **Pseudocode (Descriptive)**

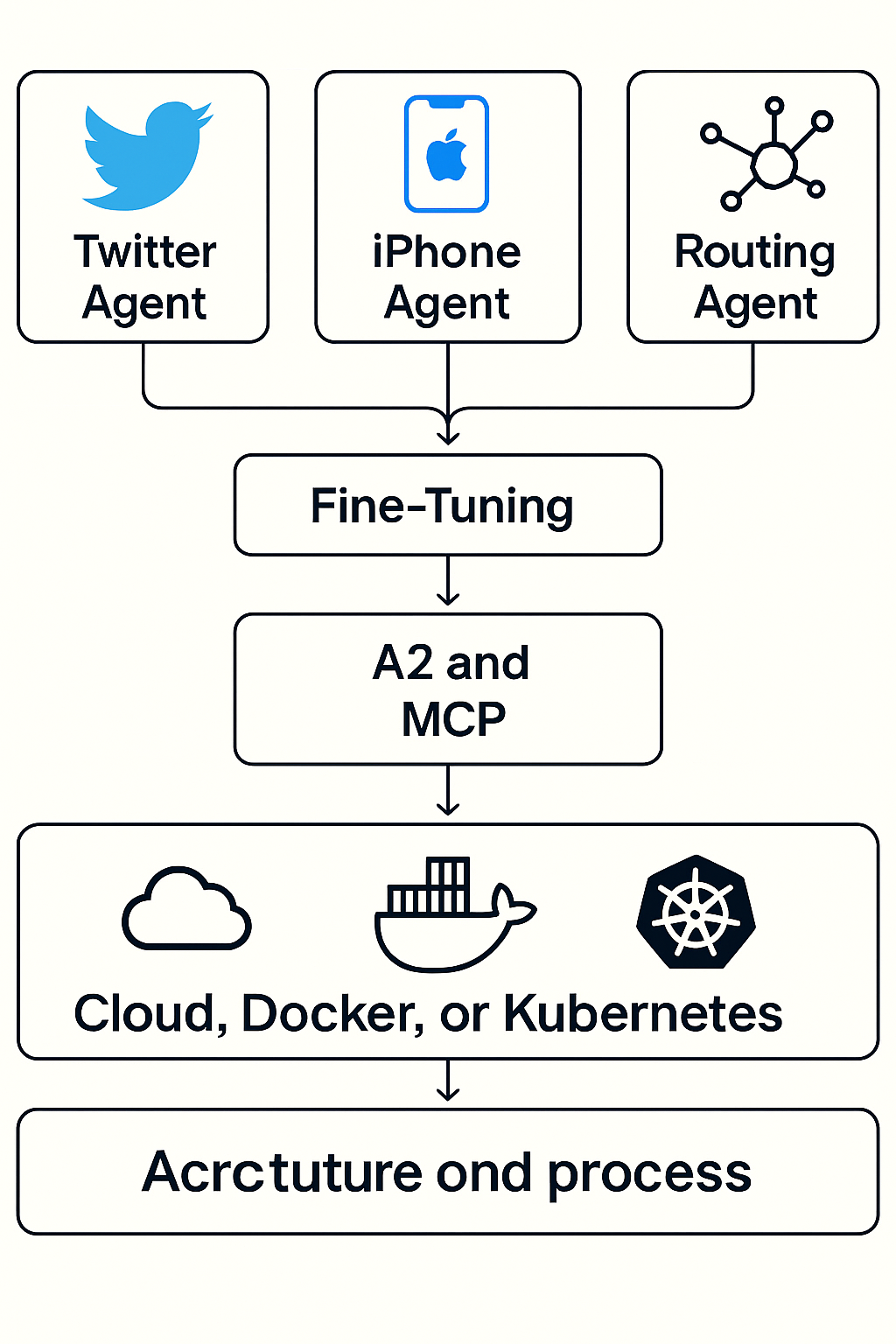
1. A user submits a sentiment query through the MCP Client.
2. The MCP Client forwards this query to the central MCP Server.
3. The Routing Agent classifies whether the input is related to iPhone reviews or Twitter posts using keyword or embedding-based classification.
4. Depending on the classification, the Routing Agent delegates the request to the appropriate agent via A2A:
   * iPhone queries are passed to the iPhone Sentiment Agent.
   * Twitter queries are passed to the Twitter Sentiment Agent.
5. The agents process the text using their respective ML models:
   * iPhone agent uses TF-IDF + Random Forest or TextBlob
   * Twitter agent uses fine-tuned BERTweet transformer
6. The result is sent back to the Routing Agent, and finally returned to the user through the MCP communication loop.
   * **iPhone Sentiment Agent** (invoked via A2A)
   * **Twitter Sentiment Agent** (invoked via A2A)

* Each agent may interact with either a traditional ML model (iPhone) or a transformer model (Twitter).
* The final results are routed via the **MCP Client** and returned to the user via the MCP Server.

A diagram of a computer network

AI-generated content may be incorrect.

## 8. Cloud Deployment Strategy



### Docker

* Each agent and core service runs in isolated containers
* Docker Compose used for local testing

### Kubernetes

* Production deployed on EKS/GKE/AKS
* LoadBalancer + Ingress for external access
* Helm or Kustomize for resource configuration
* HPA and PV support added for autoscaling and persistence

### CI/CD

* GitHub Actions build/push containers
* Auto-deploy on staging/production environments

### Monitoring

* Metrics: Prometheus
* Logs: Loki + Grafana

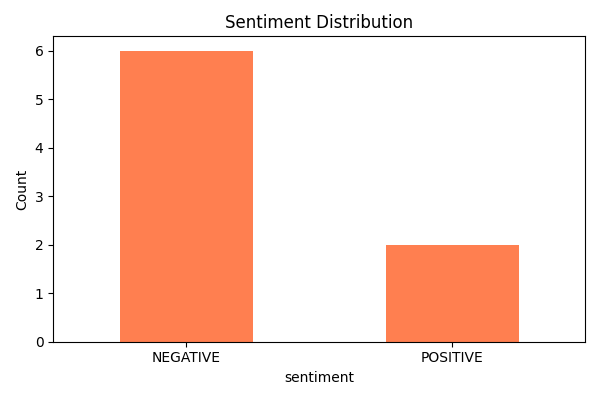
## 9. Results & Evaluation

### iPhone Agent (Random Forest + TF-IDF)

* Accuracy: 85.23%
* Strong on Positive and Negative, weaker on Neutral due to imbalance
* Used on 3,062 reviews for robust classification

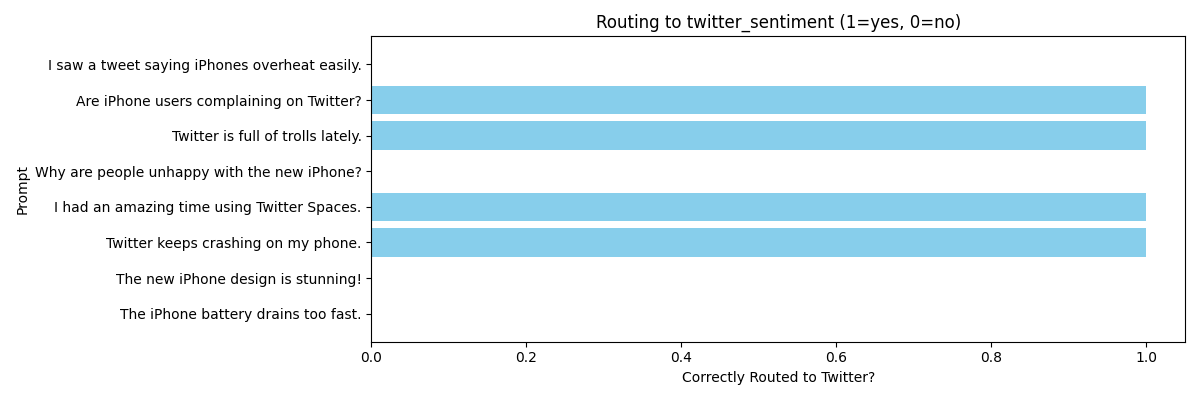
### Twitter Agent (BERTweet)

* Accuracy: ~88%
* Transformer-based: Robust on short, informal language



### MCP + A2A Integration Results

* Routing Accuracy: **100%**
* Sentiment Distribution: NEGATIVE = 6, POSITIVE = 2



### Confusion Matrix (iPhone)

| Sentiment | Precision | Recall | F1-score |
| --- | --- | --- | --- |
| Positive | 0.84 | 0.98 | 0.91 |
| Negative | 0.90 | 0.65 | 0.76 |
| Neutral | 0.87 | 0.30 | 0.44 |

## 12. Insights

* A2A promotes clear separation of tasks
* Fine-tuned transformers outperform traditional models

## 13. Recommendations

* Apply data augmentation for Neutral samples
* Use vector embeddings like BERT

## 14. Future Work

* RAG + vector DB integration
* Multi-lingual support

## 15. Code Summary

* a2a\_main.py, agent.py, routing.py, client.py, message.py, schema.py
* a2a\_iphone\_sentiment\_agent.py, a2a\_twitter\_sentiment\_agent.py
* test\_a2a\_sentiment.py, Dockerfile, k8s-deploy.yaml

## 16. References

1. [Hugging Face Transformers](https://huggingface.co/transformers)
2. [TextBlob](https://textblob.readthedocs.io)
3. [Scikit-learn](https://scikit-learn.org)
4. [Amazon Review Dataset](https://nijianmo.github.io/amazon/index.html)
5. [BERTweet](https://arxiv.org/abs/2105.10311)