FedSearch-NLP: Federated RAG QA System (Organization-Scale)

1. Project Overview

FedSearch-NLP is a large-scale, organization-level Retrieval-Augmented Question Answering (RAG) system built with Federated Learning (FL). Each department in an enterprise holds private documents (PDFs, SOPs, chats, reports). These departments train a shared global QA system collaboratively without sharing any raw text.

This system uses a centralized aggregator to merge encrypted updates, while all sensitive data stays within departments.

---

2. Core Architecture

2.1 Components per Client (Department)

- Local document preprocessing
- Local vector DB using FAISS
- Local embedding model (Sentence-Transformer / BERT-Large)
- Local adapter-based LLM for answer generation (Flan-T5-Large with LoRA)
- Local RAG pipeline
- Local training loop + DP-SGD

2.2 Central Server (Aggregator)

- Parameter aggregation (FedAvg / FedProx / SCAFFOLD)
- Handles secure aggregation
- Maintains global retriever + adapter weights

2.3 Privacy Protections

- Differential Privacy (DP-SGD)
- Update encryption
- No raw documents ever shared

---

3. Model Choices (Large, Enterprise-Level)

Retriever Model

- **all-mpnet-base-v2** or **bge-large-en** (enterprise-grade embedding models)
- Trained federatively across departments

Generator Model (LLM)

- **Flan-T5-Large** (780M) with LoRA adapters
- Only LoRA layers are trained and federated
- Base model remains frozen for speed & stability

RAG Pipeline

1. Query → client embedding
2. FAISS search (local)
3. Top-k context passed to generator
4. Generator produces answer

---

4. Workflow Summary

1. Each department loads global model.
2. Local RAG training on their private documents.
3. LoRA & retriever updates computed.
4. DP noise added.
5. Secure aggregation merges updates globally.
6. Updated global model redistributed.

---

5. Federated Algorithms Used

- **FedAvg** → baseline
- **FedProx** → best for domain drift across departments
- **Adapters-only FL** → reduces communication cost 95%

---

6. Dataset Strategy

Internal Documents (simulated)

- HR policies
- Legal docs
- IT technical SOPs
- Slack/email chat logs
- Product documentation
- Research reports

Public Sources (for base fine-tuning)

- Wikipedia dumps
- SQuAD / NQ

- MS MARCO

---

7. Evaluation Metrics

Retrieval

- Recall@5, Recall@10
- MRR

QA

- F1
- EM (Exact Match)
- ROUGE-L

Federated Metrics

- Communication cost per round
- Accuracy vs number of clients
- Privacy budget (ε)

---

8. Deployment Plan

Phase 1: Local Simulator

- Multiple simulated clients on one machine
- Flower or FedML

Phase 2: Cloud Deployment

- Docker containers (one per department)
- Load balancer for server

---

9. Code Components (that will be delivered next)

- **client.py** → preprocessing, FAISS index, local training
- **server.py** → FedAvg aggregation
- **model_retriever.py** → embedding model
- **model_generator.py** → T5 + LoRA
- **rag_pipeline.py** → query → retrieve → generate
- **federated_train.py** → full system runner
- **configs/** → hyperparameters
- **requirements.txt**

10. Final Output Deliverables

You will receive in next step:

- Complete working codebase (PyTorch + RAG + FL)
- Ready-to-run simulation
- Configurable number of departments/clients
- Trained sample model
- Startup instructions

11. Notes

- Optimized for completion in 2 days
- Minimal but powerful architecture
- Uses enterprise-level models, not small ones

12. Next Step

I will now generate the **full code** for: - server - client - local training - RAG modules - FL integration - FAISS indexing - LoRA adapters setup

Once code is integrated, I will also prepare a GitHub-ready README.