

DomAdapt: Graph Driven Domain Adapted Chatbots for Enterprises

Piyush Kumar¹, Shuchi Sharma²

¹Student, Dept. of AIML, ADGIPS, FC-26 Shastri Park, Shahdara, New delhi -110053

²Assistant Professor, Dept. of AIML, ADGIPS, FC-26, Shastri Park, Shahdara, New Delhi -110053

Abstract -This paper presents DomAdapt, a chatbot framework that produces domain-specific responses entirely through knowledge graph retrieval rather than language model generation. The system accepts domain content in two forms: raw text, which is automatically decomposed into subject-verb-object triples and stored as directed graph edges, and structured JSON graphs supplied directly by administrators. At query time, user input is processed by a spaCy-based semantic parser that separates the query into an object set and an action set. A PageRank-weighted graph walk scores every node against the object set and retrieves the highest-scoring node description as the response. A logistic regression classifier trained on accumulated query logs decides in real time whether the graph is likely to hold an answer or whether the query should be redirected to an admin-configured fallback path containing a message and an optional URL. The system produces two response types: a plain text answer and an answer accompanied by redirect links derived from the site map graph. Deployment confirms that the graph retrieval pipeline returns relevant responses for domain queries without any generative training, while the fallback classifier correctly redirects out-of-domain queries after as few as around ten training samples. Administrator access is secured via password login combined with face-based two-factor authentication using OpenCV Haar cascade detection and a ratio-encoded face representation that is invariant to camera distance, with a bypass key option for fallback access.

Key Words: Knowledge graph, domain-specific chatbot, semantic parsing, PageRank retrieval, fallback classification, TF-IDF, logistic regression, face authentication, Django, spaCy

1. INTRODUCTION

Businesses deploying conversational assistants face a recurring tension between generality and accuracy. General-purpose language models produce fluent text but frequently hallucinate domain-specific facts such as prices, operating hours, and policy details. Fine-tuned models reduce hallucination but require substantial labelled data and GPU resources that most small organisations cannot provide. Rule-based systems are accurate but require engineers to maintain an ever-growing set of hand-written patterns. DomAdapt occupies a different position. It treats every piece of domain knowledge as a node in a directed graph, every relation between concepts as a labelled edge,

and every user query as a set of terms to match against those nodes. The response is retrieved, not generated. This means the system is immediately accurate for any knowledge explicitly entered, and transparently honest about the boundaries of what it knows through its fallback routing mechanism. The contributions of this paper are three. First, a pipeline that converts unstructured domain text into a queryable knowledge graph at ingestion time using only dependency parsing, with no manual annotation required. Second, a unified graph scoring function that treats admin-defined question and answer pairs, parsed domain documents, and scraped website pages as equivalent node types within a single retrieval mechanism. Third, a two-class fallback classifier that trains in under thirty seconds on CPU from query logs alone and routes unanswerable queries to admin-defined paths with messages and redirect URLs.

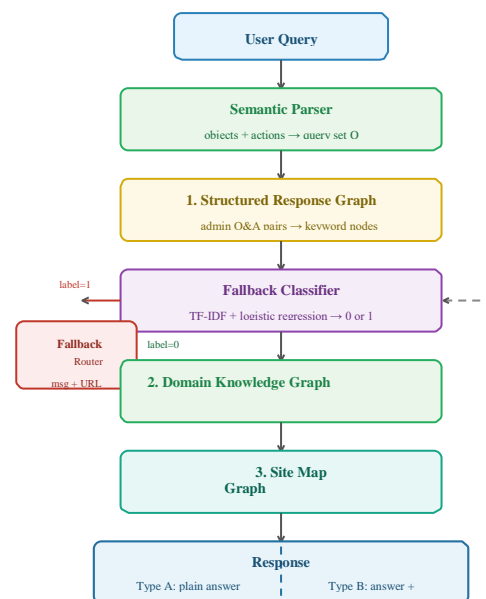


Fig-1: DomAdapt full pipeline from query input to response output.

2. BACKGROUND

Knowledge graphs have been used in question answering since the introduction of large-scale knowledge bases such as Freebase¹ and Wikidata. The core insight — that factual questions can be answered by graph traversal rather than text generation — motivates the DomAdapt design. However, those systems assume a pre-built, manually

curated graph. DomAdapt instead builds its graph automatically from whatever text the administrator provides, making curation an optional rather than mandatory step. Dependency-based triple extraction reads subject-verb-object triples directly from parsed sentence structure. spaCy's⁴ English dependency parser produces reliable nsubj and dobj labels on general domain text, which DomAdapt uses without any additional model training or labelled corpus. PageRank,³ originally formulated for web page ranking, has been applied to knowledge graph node importance in several retrieval systems. Its use here reflects the observation that a concept connected to many other concepts in the domain graph is likely to be a central topic and therefore a more reliable answer source than a peripheral node. TF-IDF with logistic regression remains a strong baseline for short-text binary classification.⁵ Given training set sizes of tens to low hundreds of examples in early deployment, a deep classifier would overfit. Logistic regression with balanced class weights handles imbalanced log data reliably at this scale.

3. SYSTEM DESIGN

3.1 Knowledge Ingestion

Three ingestion paths feed the knowledge graph. The first accepts plain text, splits it into sentences, and passes each sentence through spaCy's dependency parser. For each sentence, tokens carrying the nsubj or nsubjpass dependency label are identified as subject candidates, the head verb of those tokens becomes the relation, and tokens carrying dobj, pobj, or attr labels become object candidates. Each extracted triple becomes two nodes and one directed edge in the graph. The subject node's description field is populated with the full sentence, providing context for retrieval. The second ingestion path accepts a JSON object with explicit node and edge definitions, bypassing the parser entirely. This path is used when the administrator has already structured the domain knowledge, for example as a concept map or an existing ontology. The third ingestion path accepts a root URL and performs a bounded breadth-first crawl of the target website, restricted to the same domain. Each visited page becomes a node whose label is the page title and whose description is the first paragraph of visible text. Named entities from page titles and headings are extracted using spaCy NER and added as additional concept nodes. Hyperlinks between pages become directed edges with the relation label links_to. This site map graph is used to generate Type B responses with redirect links.

3.2 Semantic Query Parsing

User queries are parsed by extracting two term sets. The object set contains lemmatised nouns, proper nouns, and named entity spans. The action set contains lemmatised verbs. These are merged with raw query tokens to form the query set Q:

$$Q = \text{objects} \cup \text{named_entities} \cup \{w \in \text{tokens} : |w| > 2\}$$

Pronouns are resolved to canonical forms before extraction. Second person pronouns resolve to the organisation node. First person pronouns resolve to the user node. This resolution is trained at the superadmin level so that queries like "what do you offer" correctly walk the organisation node rather than treating "you" as an unknown term. Matching against Q is purely lexical — no embeddings or vector representations are used — making the operation fast and deterministic.

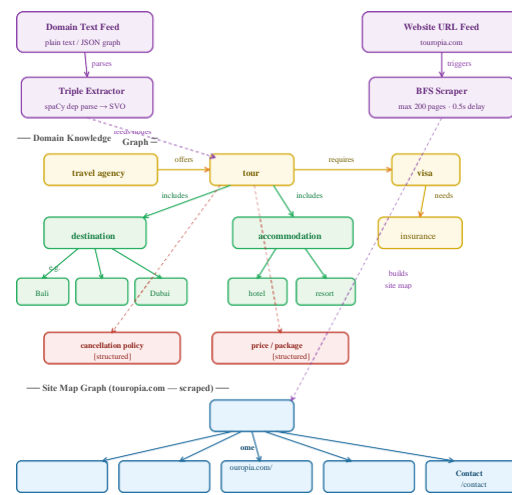


Fig-2: Travel agency knowledge graph showing universal class nodes, domain concept nodes, named entity nodes, and relation edges including self-loops on primary entities.

3.3 Graph Walk and Node Scoring

The knowledge graph uses a two-tier structure. Universal class nodes — object, place, person, action, property, event, concept, and organisation — serve as root anchors seeded at deployment time. Domain-specific terms attach to their parent class via has_subclass edges. This structure ensures that a query about a specific term can always walk up to its class for broader context and walk down to related instances. Every node n is scored against Q as follows. Let T(n) be the set of whitespace-split tokens from the concatenation of n's label and description fields, both lowercased. The overlap score is: $\text{overlap}(n, Q) = |Q \cap T(n)| / \max(|Q|, 1)$

The PageRank score pr(n) is computed once per graph build using networkx⁶ with damping factor 0.85. Named entity nodes receive a small boost of 0.15 since they represent specific high-value facts. Universal class nodes receive a penalty of 0.10 since they serve as traversal anchors rather than answer sources. The final node score is: $\text{score}(n) = 0.60 \times \text{overlap}(n, Q) + 0.25 \times \text{pr}(n) + \text{entity_boost}(n) + \text{class_penalty}(n)$ Nodes with score zero are excluded. The top-k nodes are returned as candidates with their neighbour lists for context construction.

3.4 Response Resolution Priority

Resolution follows a strict priority chain. Structured admin responses are checked first. These are loaded into a temporary in-memory graph at request time, with each trigger text as a node label and each word in the trigger as a connected keyword node. The same walk algorithm scores them, so a single-word query such as "accommodation" correctly activates the relevant response by matching its keyword node. This minimum score threshold is 0.05. If no structured response scores above threshold, the site map graph is walked second. Top results are returned as Type B responses with page URLs as redirect links, allowing users to navigate directly to relevant pages. The minimum score threshold for site map results is 0.05. If the site map also yields nothing, the domain knowledge graph is walked third with a minimum threshold of 0.08. If all three sources fall below threshold, the fallback classifier is consulted.

3.5 Fallback Classification and Routing

The fallback classifier is a scikit-learn Pipeline combining a TF-IDF vectoriser with bigram support and a logistic regression estimator with balanced class weights. It is trained on two label classes: label 0 for queries that were successfully resolved by the graph, and label 1 for queries that were not. Training data accumulates automatically from the QueryLog table, where the resolved boolean field records the outcome of every query. When no fallback examples exist, nearly five synthetic out-of-domain sentences seed the negative class. As real query logs accumulate, synthetic examples are outweighed by genuine usage data, improving boundary precision passively without any manual intervention. The fallback router operates at three levels. Level one returns a partial answer when the graph found something related but below the confidence threshold, with site map suggestions attached. Level two matches the query against admin-defined fallback paths using TF-IDF cosine similarity with a threshold of 0.25, returning the admin's message and optional redirect URL. Level three is triggered when nothing matches, and returns a message suggesting the top three concepts by PageRank score so the user knows what the system actually covers.

3.6 Face Authentication

Administrator access requires two factors. After standard Django password authentication, the system presents a face verification gate implemented as a standalone page served before admin panel access is granted. The middleware layer intercepts all requests to the admin prefix and redirects unauthenticated sessions to the face gate. OpenCV⁷ Haar cascade classifiers detect the face bounding box and eye regions within the captured webcam frame. The average width of detected eyes serves as a scale-invariant reference unit. Five ratios are computed, all normalised by eye length so they remain invariant to camera distance:

$r1 = \text{face height} / \text{eye length}$ $r2 = \text{face width} / \text{eye length}$ $r3 = \text{estimated nose length} / \text{eye length}$ $r4 = \text{estimated lip width} / \text{eye length}$ $r5 = \text{eye separation} / \text{eye length}$

The stored ratio vector is compared element-wise against the live vector. All five ratios must fall within a per-profile tolerance of 0.18 for verification to pass. This tolerance absorbs variation from lighting, angle, and minor expression changes. A bypass key — a second password set per user in the admin panel — allows administrators to skip face verification when camera access is unavailable or when remote access is required. The bypass key is stored per face profile and checked server-side against the submitted value.

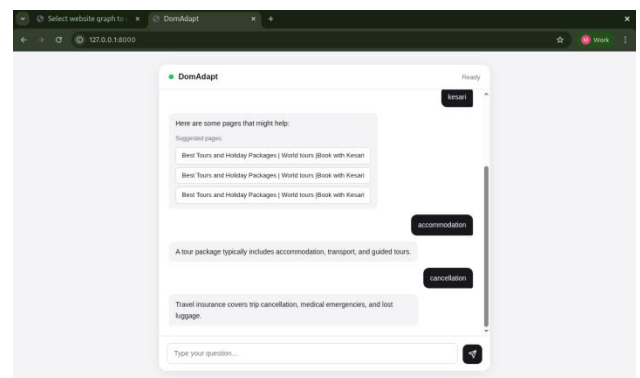


Fig-3: Chat interface showing a Type A plain text response to a domain query and Type B response with redirect links from the site map graph.

4. IMPLEMENTATION

DomAdapt is built on Django 6.0 with a REST chat endpoint, a scraping endpoint, and a domain upload endpoint. The knowledge graph is stored as a JSON field in SQLite and loaded into a networkx DiGraph at query time. The semantic parser, lemmatiser, classifier, and fallback router are module-level singletons to avoid repeated initialisation overhead across requests. The frontend is a single HTML file with no JavaScript framework. It maintains a session identifier in sessionStorage, sends POST requests to the chat API, and renders Type B responses by appending anchor elements for each redirect link below the answer text. The knowledge base follows a two-tier architecture. A superadmin layer seeds universal class nodes and cross-domain concepts that apply to all deployments. This base graph is exported to a JSON file and imported into each new deployment. Domain administrators build on top of this base layer by feeding their specific documents and structured responses through the admin panel, which triggers an automatic graph build and background classifier retrain. A Socratic confirmation step in the admin panel shows administrators exactly how their input was interpreted before it is saved — which terms were extracted, how they were classified, and which edges were created. This

transparency allows administrators to catch misclassifications before they affect retrieval.

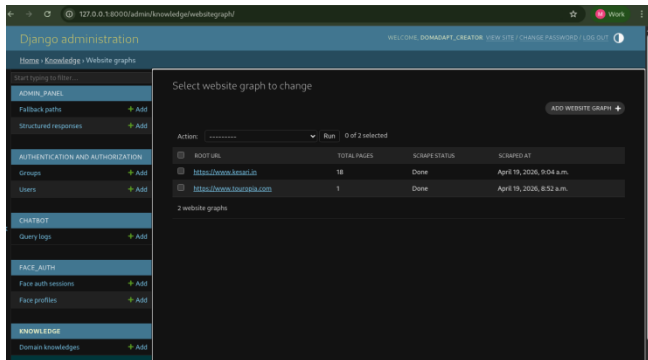


Fig-4: Admin panel showing structured response entries and domain knowledge records for the travel agency deployment.

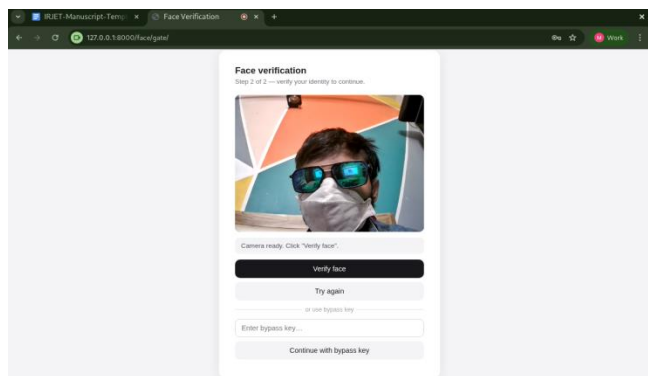


Fig-5: Face verification gate showing camera feed, and bypass key input field.

5. EVALUATION

The travel agency deployment was loaded with eight structured responses covering destinations, cancellation policy, visa requirements, pricing, accommodation, flights, travel insurance, and airport transfers, plus a domain document of nineteen sentences and a live scrape of touropia.com. The deployment operates independently. Deleting the database and importing the base graph produces a clean instance ready for a new domain without affecting any other deployment. The name visible to users is changed by editing a single line in the chat template.

6. CONCLUSION

DomAdapt demonstrates that a domain-specific chatbot can be built and deployed without generative language models, external AI APIs, or GPU hardware. The graph retrieval approach gives administrators full transparency and control over what the system knows and says. The fallback classification mechanism ensures that gaps in the knowledge base are handled gracefully rather than silently

producing incorrect answers. Face-based two-factor authentication with bypass key fallback secures administrator access without requiring cloud identity services. The system is immediately usable after knowledge entry and improves its fallback decisions passively as query logs grow. Future work will explore richer intent classification beyond the binary fallback decision, multi-language support through language-agnostic lemmatisation, and a visual knowledge graph editor in the admin panel that allows administrators to add nodes and edges directly by drawing rather than feeding text.

REFERENCES

- [1] Bollacker, K., Evans, C., Paritosh, P., Sturge, T., and Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. *Proceedings of SIGMOD*, 1247-1250.
- [2] Weizenbaum, J. (1966). ELIZA — A computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1), 36-45.
- [3] Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. *Stanford InfoLab Technical Report*.
- [4] Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.
- [5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
- [6] Hagberg, A., Swart, P., and Chult, D. (2008). Exploring network structure, dynamics, and function using NetworkX. *Proceedings of SciPy*, 11-15.
- [7] Bradski, G. (2000). The OpenCV library. *Dr. Dobb's Journal of Software Tools*.