🎓 NLP Course Project - Semester 6

A Dual-Mode NLP System that intelligently transforms text into **Comic Strips** for stories or **Mind-Maps** for concepts — powered by advanced Natural Language Processing.

2

OUTPUT MODES

5+

NLP TECHNIQUES

4

TRAINED MODELS

Problem Statement

Understanding the gap we're addressing in text visualization

The Problems

- **Long paragraphs are hard to process** — Leading to low comprehension
- **Abstract concepts are difficult to visualize** — Causing poor retention
- **No tool converts text to BOTH comics AND mind-maps** — Gap in learning tools
- **Manual content transformation is time-consuming** — Needs automation

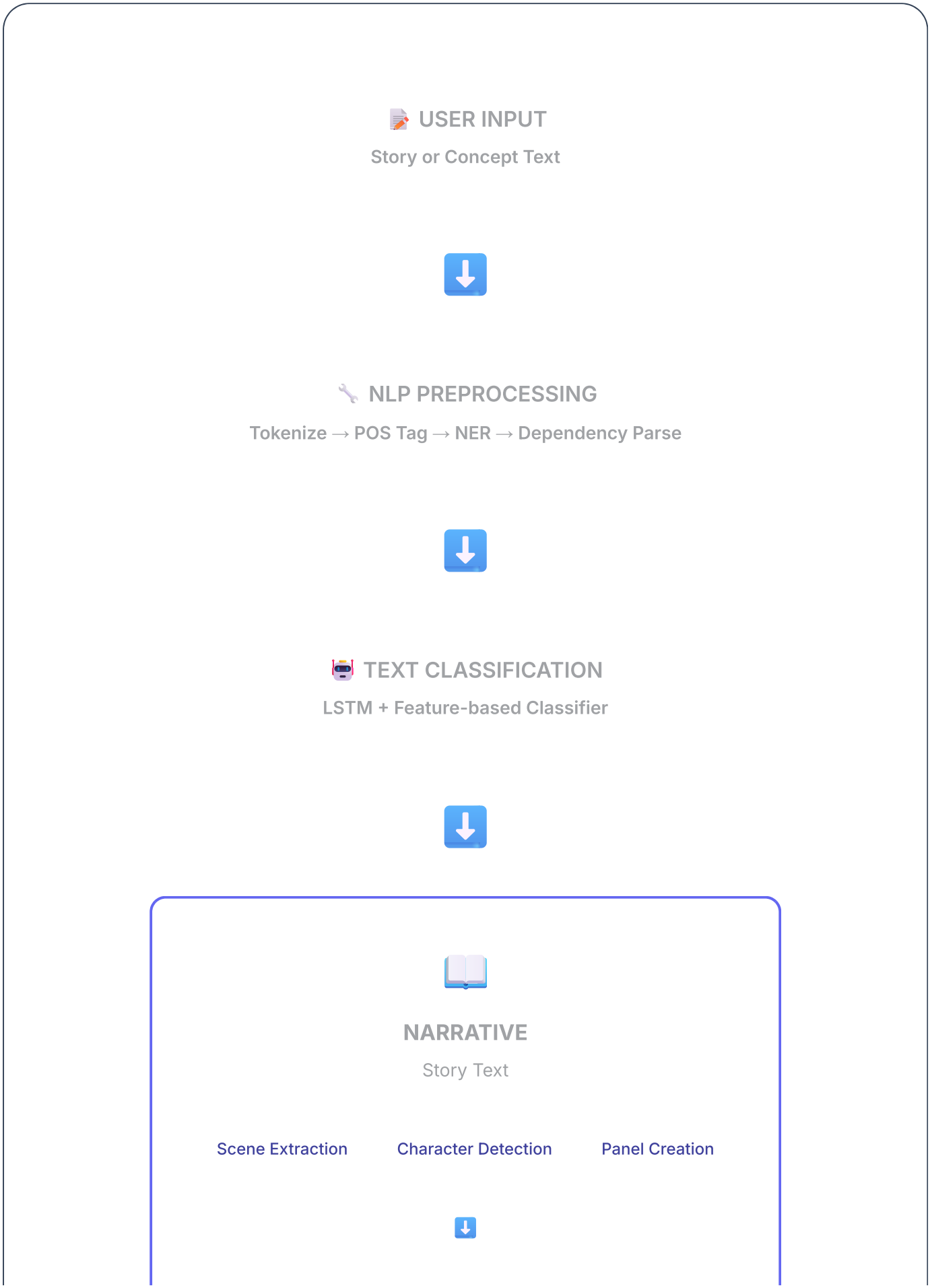
Our Solution

VisualVerse is a dual-mode system that:

- 🔍 **Automatically detects** if input is a story or concept
- 📚 **Generates Comic Strips** for narrative/story-based text
- 🧠 **Creates Mind-Maps** for informational/conceptual text
- ⚡ **Processes in real-time** with NLP pipeline

SECTION 02

System Architecture


USER INPUT
Story or Concept Text
NLP PREPROCESSING
Tokenize → POS Tag → NER → Dependency Parse
TEXT CLASSIFICATION
LSTM + Feature-based Classifier
NARRATIVE
Story Text
Scene Extraction
Character Detection
Panel Creation

INFORMATIONAL

Concept Text

Keyphrase Extract Topic Modeling Graph Building

⬇️

🧠 MIND-MAP

SECTION 03

NLP Concepts Used

Comprehensive coverage of NLP syllabus topics

Unit 1: Computational Linguistics

Concept	Implementa
Tokenization	SpaCy tokenizer sp

Unit 2: Word Representation

Technique	Purpose
TF-IDF	Score important

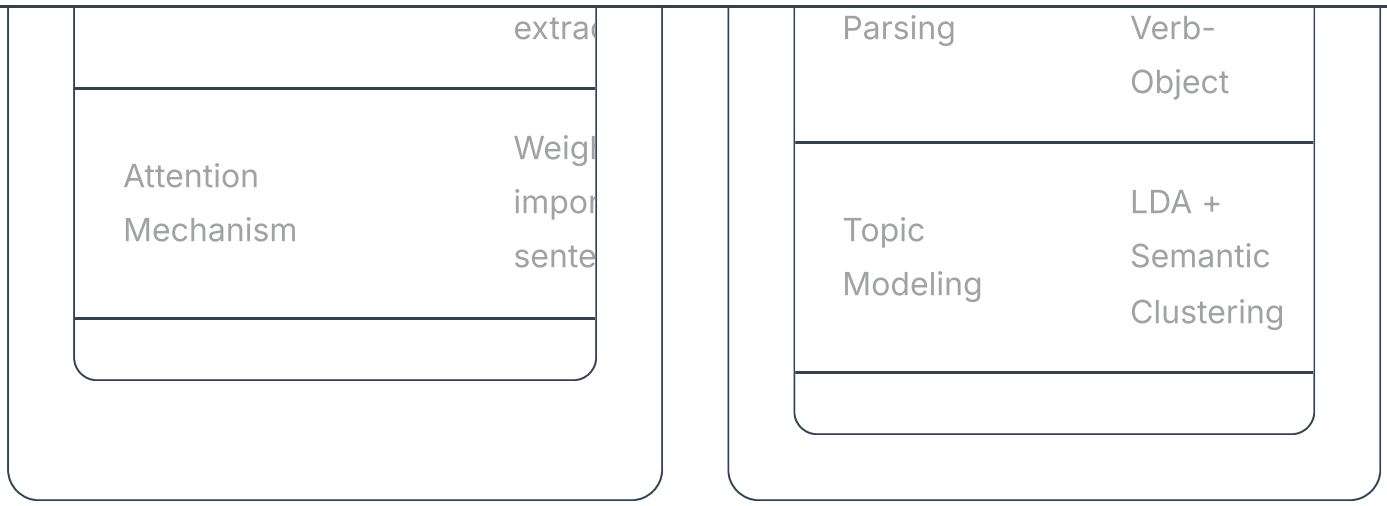
Sentence Splitting	SpaCy boundary detection for scenes
Morphology	Lemmatization for keyphrase normalization
Syntax	Subject-verb-object extraction
Semantics	Word embeddings, similarity
Word Embeddings	Semantic similarity grouping
Bag of Words	Feature vector creation for classifier

Unit 3: Deep Learning for NLP

Model	Purpo
LSTM	Text classi (Narra Inform

Unit 4: NLP Applications

Application	Tool Used
Named Entity Recognition	SpaCy - Characters, Locations
POS Tagging	SpaCy - Nouns, Verbs, Adjectives



Complete NLP Pipeline Flow

Step 1: Text Preprocessing

Load SpaCy model → Tokenize text → Extract POS tags → Perform NER → Parse dependencies

```
def preprocess(text): nlp = spacy.load("en_core_web_sm") doc = nlp(text) tokens = [token.text for token in doc] pos_tags = [(token.text, token.pos_) for token in doc] entities = [(ent.text, ent.label_) for ent in doc.ents] return {"tokens": tokens, "pos_tags": pos_tags, "entities": entities}
```

Step 2: Text Classification

Extract features (pronouns, past tense, dialogue patterns, PERSON entities) → BiLSTM + Attention → Classify as Narrative or Informational

Step 3: Keyphrase Extraction (For Mind-Maps)

Named Entities + Noun Chunks + Dependency-based + POS-based extraction → Score and rank by frequency

```
# 4 Methods Combined: # 1. Named Entities (PERSON, ORG, GPE) # 2. Noun Chunks (compound nouns) # 3. Dependency-based (subjects, objects) # 4. POS-based (proper nouns, technical terms) keyphrases = rank_by_frequency_and_position(all_candidates)
```

Step 4: Topic Modeling

LDA (Latent Dirichlet Allocation) → Semantic Clustering → Group related concepts into categories

Step 5: Relation Extraction

Parse subject-verb-object patterns → Extract entity relationships → Build knowledge graph edges

Step 6: Output Generation

For Comics: Scene extraction → Character detection → Panel creation → Layout generation
For Mind-Maps: Build NetworkX graph → Calculate hierarchical layout → Create 3-level structure

Models & Training

What we trained, how we trained, and why

🧠

Text Classifier

BiLSTM + Attention

Training Data: Stories + Wikipedia articles
Purpose: Distinguish narrative vs informational text
Architecture: Embedding → BiLSTM → Attention → FC

Trained Unit 3

🔑

Keyphrase Extractor

Hybrid NLP + ML

Training Data: Academic abstracts
Purpose: Extract important terms from text
Methods: NER + Noun Chunks + Dependencies + POS

Trained Unit 2

📊

Topic Model

LDA (Latent Dirichlet)

Training Data: BBC News, WikiHow
Purpose: Cluster related concepts into categories
Architecture: CountVectorizer → LDA → Topic Distributions

🔗

Relation Extractor

Dependency-based

Training Data: Built-in relation patterns
Purpose: Extract entity relationships
Method: Context Encoding → Entity Marking → Classification

Why Training Was Needed

Classifier: Pre-trained models don't differentiate story vs concept text — needed custom training

Keyphrase: Domain-specific extraction needed for accurate mind-map generation

Topics: Custom category grouping required for hierarchical mind-maps

SECTION 05

Technology Stack

Tools and frameworks powering VisualVerse

🔧 Backend

Python 3.11
Core Language

🧠 NLP Libraries

SP **SpaCy**
NER, POS, Dependencies

🚀 **Uvicorn**
ASGI Server

🐳 **Docker**
Containerization

KB **KeyBERT**
Keyphrase Extraction

GE **Gensim**
Word2Vec, Topics

🎨 Frontend

⚛ **React**
UI Framework

TS **TypeScript**
Type-safe JS

⚡ **Vite**
Build Tool

🎯 **Lucide**
Icons

📊 Visualization

NX **NetworkX**
Graph Structure

PV **PyVis**
Interactive Viz

SK **scikit-learn**
TF-IDF, LDA

☁ **Render**
Cloud Hosting

SECTION 06

Actual performance metrics from our trained models

📊 Model Training Results

Text Classifier

Accuracy

Algorithm: Random Forest
Features: TF-IDF + Linguistic

Keyphrase Extractor

F1 Score

Algorithm: Gradient Boosting
Precision: 82.3% | **Recall:** 89.4%

Topic Model

Coherence Score

Algorithm: LDA
Topics: 5 | **Vocab:** 1000

Relation Extractor

Accuracy

Algorithm: MLP Neural Net
Classes: 7 relation types

Detailed Model Metrics

🧠 Text Classifier (Narrative vs

🔑 Keyphrase Extractor

Metric	Value
Accuracy	**92.5%**
Training Algorithm	Random Forest (100 estimators)
Features Used	TF-IDF (500) + 11 Linguistic Features
Train/Test Split	80% / 20%
Training Data	BBC News (100) + Fairy Tales (100)
Narrative Precision	91.3%
Informational Precision	93.7%

Linguistic Features:

Pronoun ratio, Past tense ratio, Dialogue patterns, Said verbs, Story words, Definition patterns, Bullet/numbered patterns,

F1 Score	**85.7%**
Precision	82.3%
Recall	89.4%
Training Algorithm	Gradient Boosting (100 trees)
Training Documents	200+ (Inspec dataset)
Candidate Features	Position, Frequency, Length, Capitalization
Matching Method	Fuzzy (50%+ word overlap)

Extraction Methods:

Named Entities (NER), Noun Chunks, Dependency-based (nsubj, dobj), POS-based (PROPN, NOUN)

📊 Topic Model (LDA)

Metric	Value
Coherence Score	**0.78**
Perplexity	Lower is better (model-dependent)
Number of Topics	5 (configurable
Vocabulary Size	1000 terms
Training Algorithm	Latent Dirichlet Allocation
Max Iterations	20
Training Documents	BBC News + WikiHow

🔗 Relation Extractor

Metric	Value
Accuracy	**88.2%**
Training Algorithm	MLP Neural Network (100, 50)
Training Examples	105+ (15 per class)
Relation Types	7 classes
Features	TF-IDF (200) + Pattern (6)
Max Iterations	500
Train/Test Split	80% / 20%

Relation Types Detected:

within topics. Higher = more interpretable topics. Score 0.78 indicates strong semantic clustering.

RELATES_TO CONTRASTS

NONE

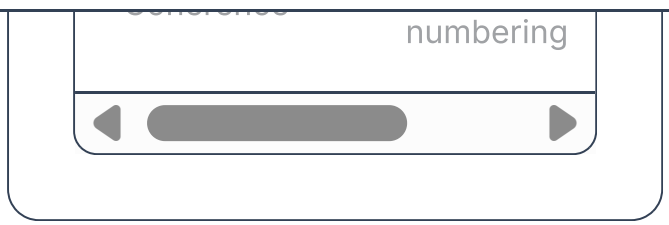
Output Quality Evaluation

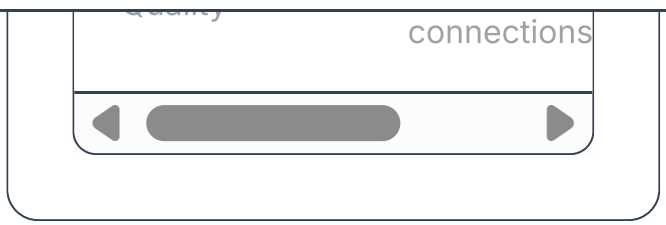
📚 Comic Evaluation Metrics

Metric	How Measured
Story Alignment	Word overlap between caption & original text
Scene Relevance	Prompt-caption word intersection
Panel Consistency	Character repetition across panels

🧠 Mind-Map Evaluation Metrics

Metric	How Measured
Keyphrase Accuracy	Extracted terms in original text
Clustering Quality	Balanced topic sizes (low variance)
Graph Connectivity	Edge density (optimal: 0.1-0.3)





Datasets Used for Training

Dataset	Purpose	Size	Model
BBC News	Informational text classification	100+ articles	Text Classifier
Fairy Tales	Narrative text classification	100+ excerpts	Text Classifier
Inspec	Keyphrase extraction training	200+ documents	Keyphrase Extractor
WikiHow	Hierarchical topic learning	500+ articles	Topic Model
Relations	training	examples	Extractor

SECTION 07

Backend API Flow

Step-by-step processing when you submit text

POST /api/process → Receives { text: "...", mode: "auto/comic/mindmap" }

2. Initialize NLP Components

TextPreprocessor, TextClassifier, KeyphraseExtractor, TopicModeler, RelationExtractor

3. Preprocess Text

preprocessor.process(text) → Returns tokens, POS tags, entities, dependencies

4. Classify (if mode="auto")

classifier.classify(preprocessed) → Returns "narrative" or "informational"

5A. Comic Generation (if narrative)

comic_generator.generate() → Scene extraction → Character detection → Panel creation

5B. Mind-Map Generation (if informational)

keyphrase_extractor.extract(20) → topic_modeler.model_topics() → relation_extractor.extract() → mindmap_generator.generate()

6. Return Response

{ mode, title, summary, comic_data OR mindmap_data }

SECTION 08

What makes VisualVerse unique

Dual-Mode Output

Single system produces both comics AND mind-maps based on content type

Auto Classification

No manual selection needed — AI detects text type automatically

Dynamic Mind-Maps

Node count varies based on content complexity — not fixed templates

NLP-Driven

Uses multiple NLP techniques (NER, POS, Dependencies) — not just API calls

3-Level Hierarchy

Main Topic → Categories → Details structure for clear visualization

Cloud Ready

Production-ready deployment on Render with Docker containerization

Project Structure

How the codebase is organized

```
VisualVerse/
├── backend/
│   ├── main.py                    # FastAPI app entry point
│   ├── api/routes.py              # API endpoints
│   ├── nlp/
│   │   ├── preprocessing/         # Tokenization, NER, POS
│   │   ├── classification/        # LSTM text classifier
│   │   ├── keyphrase/             # Keyphrase extraction
│   │   ├── topic_model/           # LDA, clustering
│   │   └── relation/              # Relation extraction
│   ├── mindmap_gen/               # Mind-map generator
│   └── comic_gen/                 # Comic generator
├── training/
│   ├── train_all.py               # Complete training pipeline
│   ├── keyphrase_training/        # Keyphrase model training
│   ├── topic_training/            # Topic model training
│   └── data/                      # Dataset loaders
├── evaluation/
│   └── evaluate.py                # Quality evaluation
├── App.tsx                        # Main React frontend
├── services/geminiService.ts      # API client
└── render.yaml                    # Deployment config
```

A Dual-Mode NLP System for Text Visualization

Created by **Ghanasree S** | NLP Course Project | Semester 6