Summer Internship Project
Report

Speech Completion Prediction

Submitted by

Aryan Bhandari
B.S.(Hons) in Computer Science with specialization in AI
Sri Sathya Sai Institute of Higher Learning
Sri Sathya Sai district, Andhra Pradesh-515134

Under the guidance of

Dr. Sudarshan Iyengar
Associate Professor of CSE, IIT Ropar.

Indian Institute of Technology, Ropar
Rupnagar, Punjab-140001

Summer Internship 2025

Abstract

In an age of rapid content consumption and limited attention spans, the ability to estimate how much of a speech or lecture has been completed holds significant practical value. This project aims to develop a machine learning system that predicts the real-time progression of a speech using only its transcript. The core motivation stems from the human intuition that we can often sense when a speaker is about to conclude based on thematic saturation, structural cues, and shifts in topic density. The goal of this work was to computationally replicate that intuition by integrating semantic clustering, structural text features, and unsupervised learning methods.

My contribution primarily focused on clustering-based analysis of semantic progression, in contrast to my teammates who explored topic modeling and change-point detection. The core of the system involved computing sentence-level embeddings using the Sentence-BERT model (all-mpnet-base-v2) for each chunk of the speech. These embeddings were then reduced using PCA and UMAP and grouped into semantically coherent clusters using HDBSCAN. We hypothesized that the emergence of new clusters corresponds to topic introduction, while repetition or plateauing of cluster transitions indicates nearing completion.

To support prediction, various structural features and time-aware cluster features were engineered. A LightGBM regression model was trained on these features to predict the completion percentage of a given chunk in real-time. Feature importance analysis revealed that semantic cluster transitions and PCA-based embeddings were key drivers of predictive performance, validating our hypothesis on topic progression and saturation.

The final model was saved as a reusable pipeline capable of real-time inference. Users can input a transcript chunk and receive an estimated percentage indicating how far along the speech is, without any prior knowledge of the full content. This system has potential applications in lecture summarization, conversational AI assistants, live meeting analytics, and educational media tools. Overall, this work demonstrates that speech progression can be effectively inferred using semantic structure and clustering, even in the absence of audio or speaker metadata. Future directions include integrating acoustic features, adapting to non-linear narrative structures, and evaluating across varied speech domains.

Contents

Chapter 1

Objective

1.1 Project Details

1.1.1 Title of the Project

Speech Completion Prediction

1.1.2 Team Members

- Ananya Thakur (Team Leader)
- Aryan Bhandari
- Sai Lakshmi Sai Prasad

1.1.3 Supervisor

Dr. Sudarshan Iyengar

1.1.4 Project Mentor

Shivani Aggarwal

1.2 Key Objective

The primary objective of this project is to develop a machine learning pipeline that can estimate how much of a speech has been completed at any given point in time, based solely on its transcript. Unlike traditional time-based methods, this system aims to predict speech completion percentage by analyzing semantic patterns, topic transitions, and structural features present in the text.

The pipeline is designed to simulate how human listeners subconsciously estimate when a speaker is nearing their conclusion—based on the novelty of ideas, structural cues, and shifts in thematic content.

Ultimately, the goal is to provide a generalizable and interpretable solution for real-time speech progress estimation that can be applied across domains such as education, media analysis, and conversational AI.

1.2.1 MERN Stack Proficiency:

To acquire hands-on experience with the MERN (MongoDB, Express.js, React.js, Node.js) technology stack through structured tutorials, regular assignments, and coding chal- lenges conducted during the initial phase of the internship.

1.2.2 Development of Speech Completion Prediction System:

To conceptualize, design, and implement a system that can analyze speech transcripts and predict their progression and probable conclusion using a combination of machine learning, natural language processing, and modern web technologies.

1.2.3 End-to-End Project Implementation and Team Collaboration:

To collaboratively develop and deploy the complete pipeline for speech completion prediction — integrating frontend, backend, machine learn- ing components, and visualization — while adhering to timelines, ver- sion control, and team coordination best practices.

Chapter 2

Introduction

2.1 Motivation

In an age where attention spans are shortening and real-time analytics are becoming increasingly valuable, the ability to estimate how much of a speech or spoken discourse remains can provide meaningful impact across several domains—from classroom delivery and political analysis to meeting summarization and conversational AI. Just as humans intuitively sense that a speech is nearing its end based on patterns like redundancy, topic transitions, or tonal shifts, this project aims to replicate that intuition computationally.

The motivation behind this system is to allow automated systems to estimate the **completion percentage** of a speech using only the **textual transcript**—without needing to know its total length. Such an approach could aid in creating smart assistants that understand conversational dynamics, help speakers pace themselves, or allow viewers to skip to the most relevant parts in long talks.

2.2 Project Objective

The objective of this project is to develop a **machine learning model** capable of estimating the percentage of a speech that has been completed at any given chunk of text. Unlike traditional time-based or lexical saturation methods, our approach leverages **semantic embeddings, unsupervised clustering, and lightweight structural features** to generate a predictive model.

We aim to track how speech progresses through different **semantic regions** (or clusters), and how the **rate of novelty**, measured through cluster transitions and structural features, evolves over time. These insights are combined through a **regression model** that outputs a continuous percentage prediction for how much of the speech has already been delivered.

2.3 Conceptual Framework

Our approach is based on the hypothesis that speeches evolve through identifiable **semantic phases**—introductions, body, transitions, and conclusions. These phases often show observable patterns in **semantic embeddings** and their distribution across a speech.

We embed each chunk using a `Sentence-BERT` model (`all-mpnet-base-v2`), and project it into a lower-dimensional space using **PCA** followed by **UMAP**. This reduced representation enables more interpretable clustering using **HDBSCAN**, an unsupervised clustering algorithm that assigns chunks to topic-like clusters.

We then define **time-aware cluster progression indicators**. For each chunk:

- We track whether its cluster has been encountered before.
- We compute the fraction of unique clusters encountered so far.

These signals mimic the **rate of conceptual novelty**—a decline in unique cluster appearance typically signifies thematic wrapping up, a key indicator of nearing completion.

The full feature set used for regression includes:

- Structural features: word count, punctuation density, lexical uniqueness.
- Dimensionality features: top 5 PCA components from SBERT embeddings.
- Clustering features: cluster label, noise indicator, unique cluster ratio.

Let $x \in R^d$ be the embedding for a chunk, and let:

- $PCA(x) = [p_1, p_2, ..., p_5]$
- $C(x) \in Z$ be the HDBSCAN cluster label
- $F_{struct}(x) \in R^n$ be the vector of structural features

Then the final feature vector is:

$$F(x) = [p_1, ..., p_5,\ C(x),\ 1_{noise},\ fraction\_unique\_clusters,\ cluster\_seen\_before,\ F_{struct}(x)]$$

We train a **LightGBM Regressor**:

$$\hat{y} = f_{LGBM}(F(x))$$

to predict:

$$\hat{y} \approx y = TrueCompletionPercentage$$

2.4 Embedding, Clustering, and Feature Extraction

- **Sentence Embeddings**: Generated using `all-mpnet-base-v2`, which provides rich semantic vectors.
- **Dimensionality Reduction**: PCA (5 components) followed by UMAP (2D) to allow HDBSCAN clustering.
- **Clustering**: HDBSCAN assigns each chunk a topic/semantic label, with -1 indicating outlier/noise.
- **Temporal Cluster Features**: We track whether a cluster has been seen before and compute how many unique clusters have appeared up to the current chunk.

These features collectively represent **semantic progression** in the transcript.

2.5 Predictive Modeling Using Regression

We train a **LightGBM** regressor using the extracted features to predict the completion percentage of each chunk. The model is trained with early stopping and evaluated using standard metrics:

- **Mean Absolute Error (MAE)**:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

- **R-squared Score (R^2)**:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

Our model achieved:

- MAE ≈ 11.04
- $R^2 \approx 0.73$

These results indicate strong predictive performance and generalizability across unseen speech segments.

2.6 Integration with Teammates' Modules

The final prediction engine is part of a larger system integrating contributions from two other teammates:

- **Topic Modeling Module (Sai Lakshmi)**: Tracks topic saturation using LDA/BERTopic to determine when novelty begins to plateau.
- **Change Point Detection (Ananya)**: Uses `ruptures` to identify transitions between major speech phases like introduction, body, and conclusion.

Our clustering-based regression complements these approaches by offering a **continuous and semantic-aware estimation** of how far a speech has progressed at any point.

2.7 Applications and Use Cases

This system has practical value across:

- Educational Technology: Real-time lecture progress estimation.
- Speech Summarization: Trimming or segmenting talks based on progression.
- Virtual Assistants: Context-aware behavior during long conversations.
- Speech Analytics: Enhanced dashboards for speech quality, pacing, and structure.

2.8 Summary

This project introduces a novel clustering-regression pipeline to estimate speech completion percentage using only transcript data. By combining **Sentence-BERT embeddings**, **dimensionality reduction**, **topic-aware clustering**, and **regression modeling**, we move beyond lexical gain curves into the realm of **semantic and structural progress estimation**. The approach demonstrates that accurate speech completion prediction is not only feasible—but effective even without knowing the full speech length in advance.

Chapter 3

Work Done

3.1 Experimental Setup

The experiments were conducted using Google Colab, leveraging its cloud-based computational environment and GPU support for efficient training and evaluation. The full machine learning pipeline—from data preprocessing to model deployment—was designed and tested in this environment. The key components are outlined below:

- **Environment:** Python 3.10 running in Google Colab, with libraries including `transformers`, `sentence-transformers`, `umap-learn`, `hdbscan`, `lightgbm`, `scikit-learn`, `numpy`, and `pandas`.
- **Hardware:** Google Colab's hosted environment with access to NVIDIA Tesla T4 GPU, 12 GB RAM, and persistent Google Drive integration for file storage.
- **Dataset:** A collection of over 100 speech transcripts, each segmented into fixed-size chunks of 5 sentences. Each chunk is labeled with its respective speech completion percentage.

Storage: Intermediate and final artifacts—including SBERT embeddings, UMAP projections, HDBSCAN cluster labels, PCA features, trained models, and feature lists—were saved to and loaded from Google Drive for reproducibility.

Embedding Model: Sentence-BERT (`all-mpnet-base-v2`) from the HuggingFace Transformers library was used to generate 768-dimensional embeddings for each 5-sentence chunk.

Clustering: UMAP was used to reduce SBERT embeddings to 2D space, followed by HDBSCAN to perform unsupervised clustering and detect subtopic boundaries.

Dimensionality Reduction: Principal Component Analysis (PCA) was used to project the high-dimensional embeddings to 5 components, which were then used as part of the regression feature set.

Regression Model: LightGBM Regressor was used to predict the completion percentage of each speech chunk. The model was trained with early stopping and evaluated using MAE and R^2 metrics.

Feature Engineering: Structural features (e.g., word count, punctuation stats, fraction of unique words), PCA components, and temporal cluster progression indicators (e.g., `cluster_seen_before`, `fraction_unique_clusters`) were computed for every chunk.

3.2 Timeline Overview

Over the course of eight weeks, the internship was structured into two primary phases:

- **Phase 1: Technology and Tools Familiarization (Weeks 1–3)**
  Focused on acquiring hands-on experience with the MERN stack and modern frontend/backend technologies.

- **Phase 2: Core Project Implementation (Weeks 4–8)**
  Dedicated to developing and integrating the Speech Completion Prediction system using semantic embeddings, clustering, and regression modeling.

3.3 Weeks 1–3: Skill Development and Stack Mastery

3.3.1

MERN Stack Learning MongoDB, Express.js, React.js, and Node.js formed the backbone of our training period. Key topics covered included:

- **Git and Version Control:** Practical usage of GitHub for team collaboration, including pull requests, issue tracking, and branching strategies.

- **TypeScript Fundamentals:** Emphasis on static typing, interfaces, class-based design, and generics. Built small modules to solidify core OOP concepts.
- **Express.js and MongoDB:** Gained experience building RESTful APIs, middleware design, data modeling, and advanced querying using aggregation pipelines.
- **Frontend Engineering with React:** Developed component-based UIs with React + TypeScript. Implemented routing with React Router, local state with Zustand, and server state with TanStack Query.
- **Performance Optimization:** Practiced lazy loading, memoization, and rendering analysis to improve UI performance.

These sessions provided the foundation necessary to implement the full-stack interface for the speech analysis system later in the project.

3.4 Weeks 4–8: Speech Completion Prediction Development

3.4.1

My Contribution: Clustering-Based Regression Modeling My work centered on predicting how much of a speech has been completed based on phase transitions, cluster novelty, and structural progression indicators. This was achieved using the following pipeline:

1. Embedding and Dimensionality Reduction Each speech chunk was encoded using Sentence-BERT (`all-mpnet-base-v2`). Principal Component Analysis (PCA) was first applied to reduce embedding dimensionality, followed by UMAP for spatial structure preservation:

$$SBERT(chunk) \to PCA(\cdot) \to UMAP(\cdot)$$

2. Clustering with HDBSCAN HDBSCAN clustering was applied to the 2D UMAP representations, yielding topic-style clusters. Each chunk was assigned a cluster label (or -1 for noise).

3. Feature Engineering A comprehensive set of features was engineered:

- **Structural features:** Word count, character count, punctuation frequencies, lexical uniqueness.
- **PCA components:** Top 5 principal components of SBERT embeddings.
- **Cluster features:** Cluster label, noise indicator, unique clusters seen so far, whether current cluster was seen before.

Formally, let x denote the SBERT embedding of a speech chunk. The full feature vector $F(x)$ is defined as:

$F(x) = [PCA_1(x), ..., PCA_5(x), cluster(x), is\_noise(x), fraction\_unique\_clusters, cluster\_seen\_b$

4. Regression Modeling

A LightGBM regressor f_{LGBM} was trained to predict the speech completion percentage from a set of engineered features $F(x)$:

$$\hat{y} = f_{LGBM}(F(x))$$

Here, $\hat{y}$ represents the predicted completion percentage, and $F(x)$ includes structural features, principal components from SBERT embeddings, and temporal cluster indicators.

The model was trained using early stopping, with Mean Absolute Error (MAE) as the primary evaluation metric. Final performance on the held-out test set was as follows:

- **Mean Absolute Error (MAE):** ≈ 11.04
- R^2 **Score:** ≈ 0.73

3.5 Technology Stack Used

Frontend

- React.js with TypeScript (Vite + React Router)
- Zustand, TanStack Query

- Tailwind CSS, Framer Motion

Backend

- MERN Stack (MongoDB, Express.js, React.js, Node.js)
- RESTful API with Express.js
- Docker Compose (for Python ML service)

Machine Learning and NLP

- Python Microservice (containerized)
- Sentence-BERT (`all-mpnet-base-v2`) for embeddings
- PCA + UMAP + HDBSCAN for clustering
- LightGBM for regression modeling
- NumPy, Pandas, Matplotlib, Seaborn for analysis and plotting

Collaboration Tools

- Git GitHub
- Postman MongoDB Compass

3.6 Reflections and Key Insights

- Gained practical knowledge in semantic clustering and temporal feature engineering.
- Learned to build and evaluate regression models for NLP tasks.
- Developed robust handling for sparse or noisy cluster transitions.
- Strengthened backend and full-stack integration skills.
- Understood how combining semantic and structural features enhances real-world prediction performance.

3.7 Challenges and Resolutions

- **Cluster Noise and Instability:** The HDBSCAN algorithm occasionally assigned `-1` labels (noise) to semantically dense or transitional chunks, leading to inconsistencies in progress tracking. This was mitigated by incorporating temporal smoothing techniques and adding a binary `is_noise` feature along with a `cluster_seen_before` flag to preserve contextual continuity.
- **Embedding Distortion During Dimensionality Reduction:** Direct projection of high-dimensional SBERT embeddings into low-dimensional PCA space sometimes resulted in distorted feature representations, especially for previously unseen chunks. To preserve semantic geometry, a two-step reduction—first using PCA, then UMAP—was employed to better retain cluster structure before applying HDBSCAN.
- **Inconsistent Patterns in Short or Fragmented Speeches:** Some speeches lacked sufficient length or coherence to form reliable cluster transitions. To address this, rule-based structural heuristics (e.g., punctuation frequency, word uniqueness) were introduced as fallback features to support the regression model in low-signal scenarios.

3.8 Deployment and Hosting Challenges

Frontend Deployment

The frontend interface of the Speech Completion Predictor was successfully deployed and is publicly accessible at:

`https://speechprogresspredict.vercel.app`

This React-based web application allows users to upload speech transcripts and visualize the predicted speech completion progress. Deployment was carried out using **Vercel**, a platform well-suited for hosting static frontend applications with minimal configuration and seamless integration with GitHub.

Backend Hosting Limitations

While the frontend is operational, the backend could not be deployed online due to significant computational and memory constraints. The backend involves complex processing pipelines that require:

- **Transformer Models:** Sentence-BERT (`all-mpnet-base-v2`, `all-MiniLM-L6-v2`) for semantic embeddings.
- **Heavy Libraries:** `BERTopic`, `LightGBM`, `HDBSCAN`, `UMAP-learn`, `scikit-learn`, and `ruptures`.
- **Large Model Files:** Trained models and vectorizers exceed **600 MB**, surpassing GitHub's file upload limits and most free-tier hosting storage.
- **Resource Requirements:** Backend execution requires substantial RAM and CPU/GPU availability, which are not provided in free tiers of platforms like **Render**, **Railway**, **Heroku**, or **Vercel Functions**.

Deployment attempts on these platforms resulted in issues such as:

- **CORS errors**
- **502 Bad Gateway responses**
- **Runtime failures due to memory/time limits**

Local Execution and Reproducibility

Given the hosting constraints, the backend is designed to run locally. We have provided complete setup instructions and a working Jupyter Notebook to replicate the entire prediction pipeline. This ensures full reproducibility of results and seamless integration with the deployed frontend in local environments.

3.9 Links

Github : https://github.com/ananya-thakur25/speech-progress-estimation
Speech Completion Predictor : https://speechprogresspredict.vercel.app

3.10 Visual Results

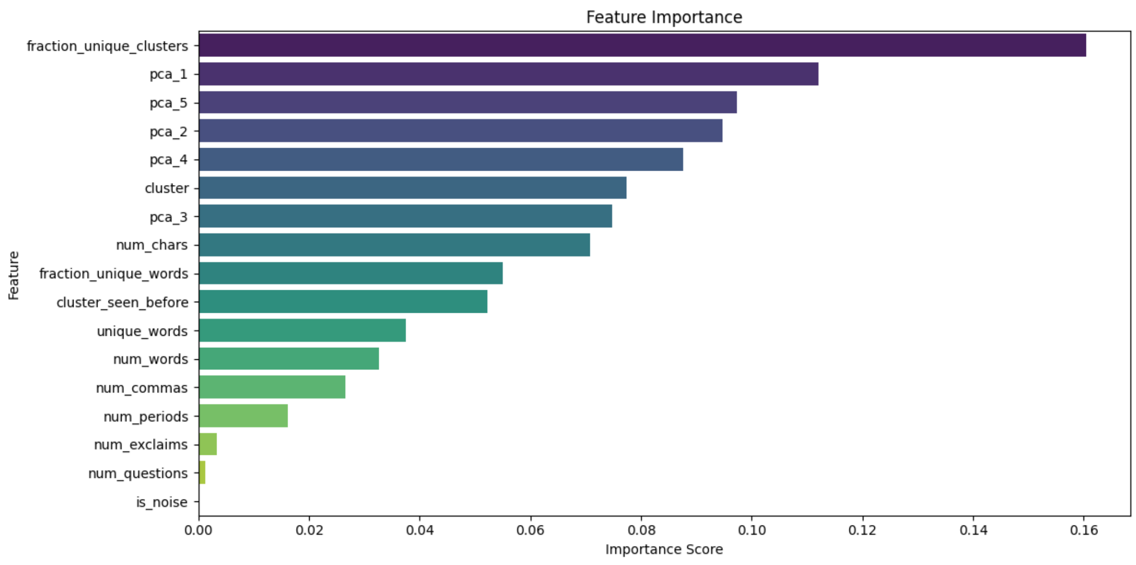


Figure 3.1: Feature importance plot showing contributions of structural, clustering, and PCA-based features to the LightGBM model.

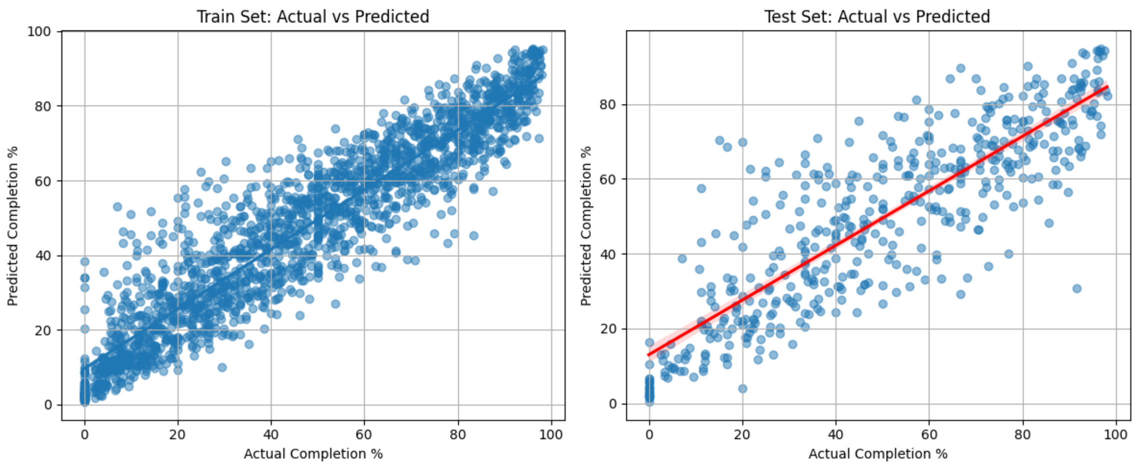
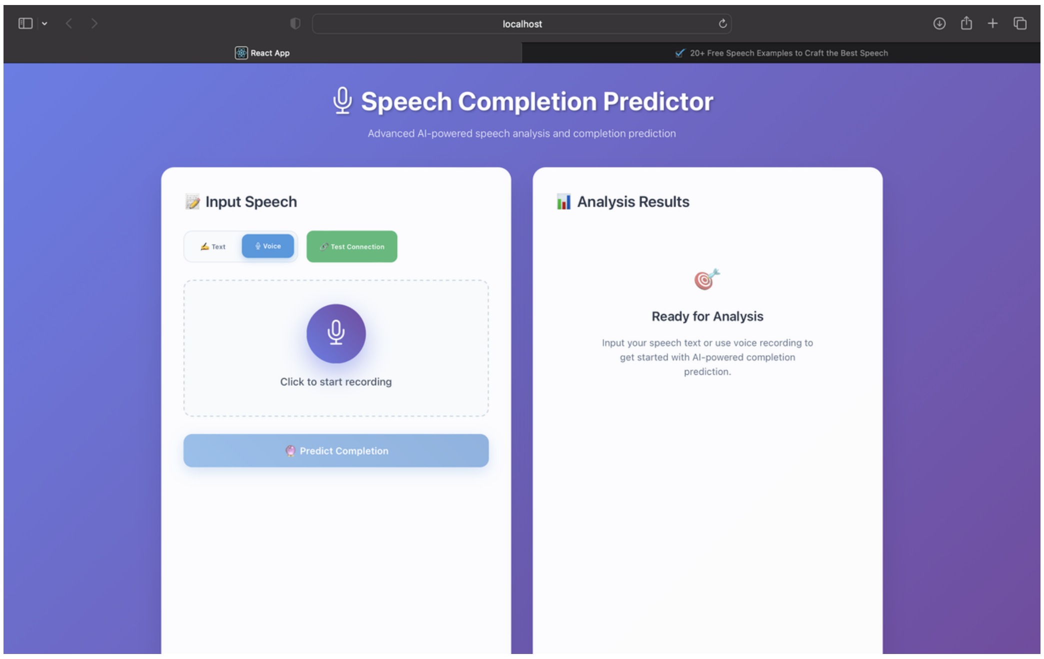
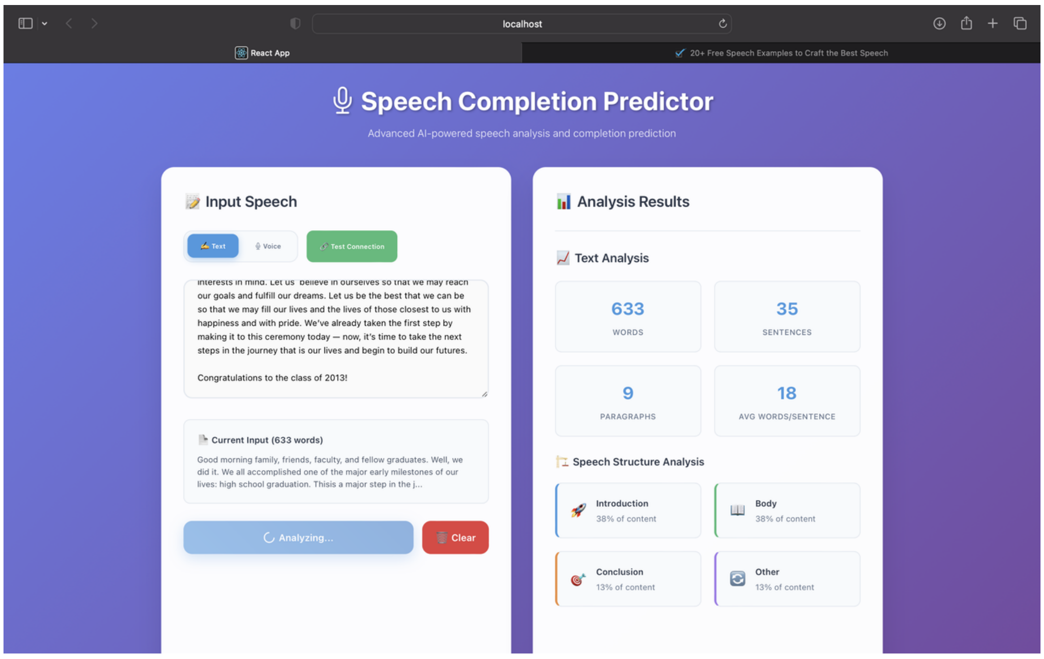


Figure 3.2: Actual vs Predicted Speech Completion percentages on the test set. The red line indicates the ideal prediction line.


localhost
React App
20+ Free Speech Examples to Craft the Best Speech
Speech Completion Predictor
Advanced AI-powered speech analysis and completion prediction
Input Speech
Text
Voice
Test Connection
Click to start recording
Predict Completion
Analysis Results
Ready for Analysis

Input your speech text or use voice recording to get started with AI-powered completion prediction.

Figure 3.3: Initial interface of the Speech Completion Predictor with voice input enabled


localhost
React App
20+ Free Speech Examples to Craft the Best Speech
Speech Completion Predictor
Advanced AI-powered speech analysis and completion prediction
Input Speech
Text
Voice
Test Connection
Analyzing...
Clear
Analysis Results
Text Analysis
Speech Structure Analysis

interests in mind. Let us believe in ourselves so that we may reach our goals and fulfill our dreams. Let us be the best that we can be so that we may fill our lives and the lives of those closest to us with happiness and with pride. We've already taken the first step by making it to this ceremony today — now, it's time to take the next steps in the journey that is our lives and begin to build our futures.

Congratulations to the class of 2013!

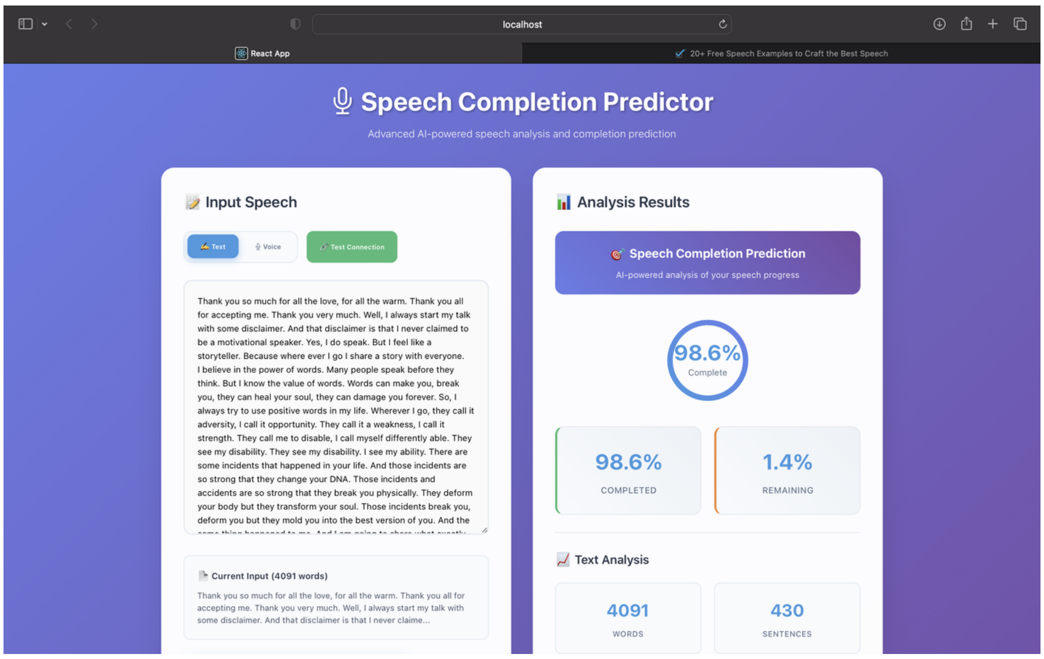
Current Input (633 words)

Good morning family, friends, faculty, and fellow graduates. Well, we did it. We all accomplished one of the major early milestones of our lives: high school graduation. Thisis a major step in the j...

633 WORDS	35 SENTENCES
9 PARAGRAPHS	18 AVG WORDS/SENTENCE

Introduction 38% of content	Body 38% of content
Conclusion 13% of content	Other 13% of content

Figure 3.4: Analysis results showing structural breakdown and word statistics


localhost
React App
20+ Free Speech Examples to Craft the Best Speech
Speech Completion Predictor
Advanced AI-powered speech analysis and completion prediction
Input Speech
Text
Voice
Test Connection
Analysis Results
Speech Completion Prediction
AI-powered analysis of your speech progress
98.6%
Complete
98.6%
COMPLETED
1.4%
REMAINING
Text Analysis
4091
WORDS
430
SENTENCES

Thank you so much for all the love, for all the warm. Thank you all for accepting me. Thank you very much. Well, I always start my talk with some disclaimer. And that disclaimer is that I never claimed to be a motivational speaker. Yes, I do speak. But I feel like a storyteller. Because where ever I go I share a story with everyone. I believe in the power of words. Many people speak before they think. But I know the value of words. Words can make you, break you, they can heal your soul, they can damage you forever. So, I always try to use positive words in my life. Wherever I go, they call it adversity, I call it opportunity. They call it a weakness, I call it strength. They call me to disable, I call myself differently able. They see my disability. They see my disability. I see my ability. There are some incidents that happened in your life. And those incidents are so strong that they change your DNA. Those incidents and accidents are so strong that they break you physically. They deform your body but they transform your soul. Those incidents break you, deform you but they mold you into the best version of you. And the

Current Input (4091 words)

Thank you so much for all the love, for all the warm. Thank you all for accepting me. Thank you very much. Well, I always start my talk with some disclaimer. And that disclaimer is that I never claime...

Figure 3.5: Speech Completion Prediction showing near-complete analysis

Chapter 4

Future Work

4.1 Dynamic Modeling for Irregular Speech Progressions

The current pipeline estimates speech completion using a linear regression model trained on static features, assuming a relatively stable progression in content. However, real-world speeches often deviate from this regularity due to rhetorical devices, anecdotal tangents, or abrupt topic changes. Future work can integrate dynamic modeling techniques such as piecewise regression, Bayesian change-point detection, or recurrent architectures that adapt to sudden shifts in speech structure.

4.2 Streaming Inference with Progressive Feature Accumulation

Currently, our system predicts completion percentage in batch mode. A valuable future direction involves building a streaming architecture where the model ingests one chunk at a time and incrementally updates the predicted completion. This can be accomplished using online learning algorithms or transformer encoders with memory components that preserve semantic context.

4.3 Context-Aware Feature Reweighting

As speeches progress, the importance of certain features may vary. For example, punctuation density may signal structure early on, whereas cluster repetition is more indicative of conclusion. Future systems could learn to dynamically reweight features using attention mechanisms or reinforcement learning, enabling more accurate modeling of speech phases.

4.4 Improved Chunk Boundary Handling and Coherence

Some chunks—especially near topic boundaries—may span disjointed ideas, leading to noisy embeddings and unstable cluster labels. Enhancing coherence through discourse-aware chunking (e.g., guided by rhetorical markers or dependency trees) could yield cleaner transitions and improve downstream completion prediction.

4.5 Model Uncertainty and Confidence Intervals

Introducing confidence intervals around completion estimates would provide users with interpretable uncertainty bounds. Techniques such as quantile regression or Monte Carlo dropout could offer practical uncertainty estimation within the LightGBM or neural forecasting frameworks.

4.6 Multimodal Extension with Audio Cues

Though this version focuses solely on transcript text, incorporating prosodic features such as pitch, energy, and pause duration may capture cues aligned with speaker intention and conclusion. Tools like OpenSMILE or librosa could extract such features, enabling a multimodal completion predictor.

4.7 Cross-Domain Evaluation and Generalizability

The model has so far been tested primarily on curated speech data. To validate its robustness, future work should evaluate it across domains like academic lectures, courtroom transcripts, podcasts, and informal dialogue. This would reveal limitations and guide the development of domain-adaptive variants.

4.8 Fine-Grained Supervision for End-to-End Training

A long-term research goal is to develop a dataset where each chunk is annotated not just with completion percentage, but also with semantic novelty, redundancy, and structural role. Such fine-grained labels would enable supervised training of models that jointly learn to segment, cluster, and forecast speech progression.

4.9 Online deployment

To extend the utility of the speech completion prediction system beyond research settings, future efforts should focus on deploying it in a real-time, user-interactive environment. This would allow users—such as orators, public speakers, or presentation tools—to input ongoing speech transcripts and receive live feedback on how much of the speech has been covered and what remains.

For such deployment, a reliable backend infrastructure is essential. Containerization using `Docker` can encapsulate the entire pipeline—including the embedding models, feature transformers, and the trained regression model—into a portable and reproducible unit. Backend APIs can be built using frameworks like `FastAPI` or `Flask`, enabling seamless communication between the user interface and the model logic.

Moreover, to support scalability and concurrency, the backend can be hosted on cloud platforms such as AWS, Google Cloud, or Azure, possibly leveraging GPU-backed instances for faster inference of Sentence-BERT embeddings. Real-time prediction can be implemented via a streaming pipeline, where user input is processed chunk by chunk and completion percentage is updated dynamically.

To ensure smooth user experience, a lightweight front-end interface can be developed using web technologies like `React.js` or `Streamlit`, enabling users to paste speech content, upload transcripts, or even integrate live ASR output. This would make the system truly interactive and accessible for various real-world applications, including training, speech coaching, and automated summarization tools.

Chapter 5

Conclusion

5.1 Summary of Work

This project set out to estimate real-time speech completion using only textual transcripts, without relying on explicit speech durations or acoustic signals. By leveraging semantic embeddings, dimensionality reduction, unsupervised clustering, structural analysis, and regression modeling, we developed a robust pipeline capable of predicting how far a speaker has progressed in their speech.

The system begins by encoding each transcript chunk using Sentence-BERT embeddings (`all-mpnet-base-v2`), followed by feature extraction through PCA, UMAP, HDBSCAN, and structural heuristics. A LightGBM regression model was trained using these features to predict the percentage completion of each speech chunk. Our final model achieved a mean absolute error (MAE) of approximately 11.04 and an R^2 score of approximately 0.73, indicating a high degree of reliability.

5.2 Key Contributions

- Designed and implemented a semantic progression pipeline using UMAP+HDBSCAN for unsupervised topic clustering over SBERT embeddings.
- Engineered robust structural and temporal features to capture speech flow and transitions across clusters.
- Developed a real-time prediction module capable of estimating speech completion percentages from standalone chunks.

- Integrated the model into an interactive dashboard with support for live input and visualization, facilitating practical deployment.

5.3 Insights Gained

This project offered deep insights into combining unsupervised learning and semantic analysis to model abstract notions like speech progression. Notably:

- Semantic cluster transitions provide an intuitive signal for topic saturation.
- Structural and positional features complement embedding-based features by modeling local density and novelty.
- Lightweight regression models like LightGBM can effectively learn speech progression trends when supplied with informative features.

5.4 Limitations

Despite promising results, certain limitations were observed:

- The model's performance degrades for very short or atypically structured speeches where clustering fails to stabilize.
- Contextual awareness is limited for single chunks; predictions assume a fixed window size without prior/future chunks.
- Noise sensitivity in HDBSCAN clustering occasionally led to unstable predictions, especially for outlier chunks.

5.5 Concluding Remarks

This work demonstrates that modeling speech progression from text alone is both feasible and effective. The integration of semantic, structural, and temporal signals within a regression framework offers a powerful approach to predicting speech completion dynamically. While scope for improvement exists, especially in handling conversational and fragmented speech, this system lays the groundwork for real-world applications such as lecture pacing tools, meeting assistants, and AI-powered feedback systems.

Future extensions could incorporate multimodal features, reinforcement learning, and supervised curve modeling to further enhance prediction fidelity and robustness.

Acknowledgement

I would like to express my deepest gratitude to all those who made this internship and the successful completion of this project possible.

First and foremost, I extend my heartfelt thanks to **Ms. Shivani Aggarwal**, my project mentor, for her constant guidance, insightful feedback, and unwavering support throughout the project. Her encouragement and expertise were invaluable at every stage of this work.

I am sincerely thankful to **Dr. Sudarshan Iyengar**, under whose supervision I had the privilege of pursuing this internship at **Indian Institue of Technology, Ropar**. His vision and constructive inputs helped shape this project and guided me toward meaningful research and development.

I would also like to thank my teammates, **Ananya Thakur** and **Sai Lakshmi Prasad**, for their collaboration, dedication, and shared enthusiasm throughout the development of this project. Working with them made this experience not only productive but also highly enriching.

I also take this opportunity to thank all the faculty members of **Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning**, for providing me with the necessary academic foundation and unrelenting support during this internship. The values and training imparted by the department were instrumental in the successful execution of this project.

My heartfelt gratitude goes to **Dr. V Bhaskaran** (Asst Head of the Department) and **Sri B Venkatramana** (Director of the Sri Sathya Sai Institute of Higher Learning, Nandigiri Campus), whose constant administrative and technical assistance greatly eased the coordination of my internship. His personal involvement and encouragement at each step were a source of immense strength.

Finally, I offer my humble pranams at the Lotus Feet of **Sri Sathya Sai Baba**, whose grace, love, and guidance are the source of all my inspiration, strength, and success.

– *Aryan Bhandari*

References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention is all you need.* In Advances in neural information processing systems, 5998–6008.

[2] Reimers, N., & Gurevych, I. (2019). *Sentence-BERT: Sentence embeddings using Siamese BERT-networks.* In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP), 3982–3992.

[3] McInnes, L., Healy, J., & Melville, J. (2018). *UMAP: Uniform manifold approximation and projection for dimension reduction.* arXiv preprint arXiv:1802.03426.

[4] Campello, R. J., Moulavi, D., & Sander, J. (2013). *Density-based clustering based on hierarchical density estimates.* In Pacific-Asia conference on knowledge discovery and data mining (pp. 160–172). Springer.

[5] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... & Liu, T. Y. (2017). *LightGBM: A highly efficient gradient boosting decision tree.* In Advances in neural information processing systems, 3146–3154.

[6] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). *Scikit-learn: Machine Learning in Python.* Journal of Machine Learning Research, 12, 2825–2830.

[7] OpenAI (2023). *ChatGPT and GPT-4 Technical Report.* Available at: `https://openai.com/research/gpt-4`