PROJECT TITLE: Contextual Disambiguation of Ambiguous Entities in Text

HARIHARAN R - 23011101104
R SENTHALIR - 23011101108

Analysis of Challenges

Textual data is inherently complex and poses several challenges, particularly when attempting to derive meaningful context. The following are key issues encountered:

Ambiguity
Many words in English have multiple meanings depending on their usage in a sentence. For instance, "Amazon" can refer to an e-commerce company or a river in South America. Without understanding context, it is easy for models to misclassify such terms.

Polysemy and Synonymy
Words may have the same or multiple meanings. For instance, “bank” might imply a river bank or a financial institution. “Car” and “automobile” are synonyms, which models need to interpret correctly to avoid redundancy.

Preprocessing Noise
While preprocessing (removing punctuation, lowercasing, etc.) helps with cleaning, it may sometimes eliminate significant tokens, especially if not carefully executed.

Linguistic Diversity
Different styles, dialects, abbreviations, or misspellings may reduce the generalization capability of models.

Contextual Complexity
Especially in resumes, where structure is not uniform, understanding context becomes extremely tricky.

Application Selection

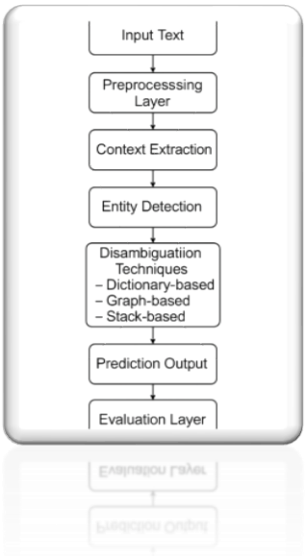
The selected application for this project is Named Entity Disambiguation through Contextual Word Sense Disambiguation. This application is vital in:

- Resume parsing
- Search engine optimization
- Document summarization
- Virtual assistants and chatbots

We implemented three disambiguation approaches:
1. Dictionary-based (Hash Map)
2. Graph-based (BFS)
3. Stack-based (Rule-based with LIFO)

Architecture Diagram

Architecture Flow:
1. Input Text – Raw sentences with ambiguous entities.
2. Preprocessing Layer – Cleaning, tokenization, stopword removal.
3. Context Extraction – Identify the surrounding context window (5 words).
4. Entity Detection – Locate ambiguous terms using an internal dictionary.
5. Disambiguation Techniques – Apply:
   - Dictionary-based
   - Graph-based
   - Stack-based
6. Prediction Output – Disambiguated entity tags.
7. Evaluation Layer – Measure accuracy using ground-truth comparison.



Module Description

Dictionary-Based Disambiguation:
- Uses a pre-defined dictionary of entities and their keyword contexts.
- Compares surrounding words with keywords to score and infer meanings.
- Accuracy: 61.33%

Graph-Based Disambiguation:
- Constructs a graph and uses BFS to score contextual proximity.
- Accuracy: 92%

Stack-Based Disambiguation:
- Uses a stack to remember recent context.
- Accuracy: 38%

Data Selection and Preprocessing

Custom data was generated including ambiguous entities. Preprocessing steps involved:

- Lowercasing
- Removing punctuation
- Stopword removal
- Tokenization

All three algorithms were implemented using Python.

Performance Evaluation

Method	Average Accuracy
Dictionary-based	61.33%
Graph-based (BFS)	92.00%
Stack-based	38.00%

Conclusion and Future Work

Conclusion:
- Dictionary-based methods are useful for lightweight solutions.
- Graph-based traversal using BFS is optimal for accurate disambiguation.
- Stack-based rules are fast but less accurate.

Future Work:

- Integrate BERT embeddings for context understanding.
- Apply on larger datasets.
- Combine methods for hybrid performance.