

Word Sense Disambiguation using Indo WordNet

Prathibha Bharathi S D

Date 09-01-2024

ABSTRACT

Polysemous Words can have more than one distinct meaning. Word sense disambiguation (WSD) is the ability to identify the exact meaning of such polysemous words in context in a computational manner. WSD is considered as an AI-complete problem, that is, a task whose solution is at least as hard as the most difficult problem in Artificial Intelligence.

WSD is viewed as an essential and also a useful process in most of the language technology applications like machine translation, information extraction or retrieval, knowledge acquisition or meaning, lexicography, and semantic interpretation, and is now even more important in latest research areas like bioinformatics and Semantic Web. Knowledge and corpus-based methods are usually applied for disambiguation task.

Keywords: Word sense disambiguation, Polysemy, IndoWordNet, Lesk Algorithm, Natural language processing.

Problem Statement

In natural language processing, word sense disambiguation (WSD) is the problem of determining which "sense" (meaning) of a word is activated by the use of the word in a particular context, a process which appears to be largely unconscious in people. WSD is a natural classification problem: Given a word and its possible senses, as defined by a dictionary, classify an occurrence of the word in context into one or more of its sense classes. The features of the context (such as neighbouring words) provide the evidence for classification. Problem statement is to develop a WSD system for Kannada language using IndoWordNet and Lesk Algorithm. which will be able to determine the most relevant sense of the word in the given context. We try to solve the problem by comparing the context of the given sentence with the gloss and example of the target word in the IndoWordNet.

Market/Customer Need Assessment

The language possesses words that have polysemous meanings which refer to words having two or more meanings. It is important to identify and disambiguate such words by machines so as to reduce the language barriers in machine learning. The machine should be enabled to process information in the language independent manner.

WSD contributes to more accurate text summarization by ensuring that the chosen words and phrases accurately represent the intended meaning within the given context.

Ambiguity in language is a common challenge, and WSD helps reduce ambiguity by selecting the most appropriate meaning for a word in a given context. This is crucial for avoiding misunderstandings and misinterpretations.

This is particularly important in applications like chatbots, virtual assistants, and sentiment analysis where context plays a crucial role in understanding user input. Aim is to develop a WSD system for Kannada language using IndoWordNet and Lesk Algorithm. which will be able to determine the most relevant sense of the word in the given context.

Target Specification and characterization

Search Engines:

Improve search engine accuracy by disambiguating words in search queries and delivering more relevant results.

Chatbots and Virtual Assistants:

Enhance the understanding of user queries in chatbot interactions, providing more accurate and context-aware responses.

Content Creation and Writing Tools:

Assist writers by disambiguating words and suggesting the most appropriate terms based on context.

Legal Document Analysis:

Assist legal professionals in analysing and understanding complex legal documents with industry-specific language.

Healthcare Documentation:

Improve accuracy in processing medical texts and documents, where ambiguity can have critical implications.

Government and Policy Analysis:

Assist in analysing policy documents, legal texts, and government communications for accurate interpretation.

Human-Machine Interfaces:

Improve interactions in human-machine interfaces by understanding natural language commands and queries.

External Search

[1] Shashank N S, Dr. Jagadish S Kallimani”Word Sense Disambiguation of Polysemy Words in Kannada Language”

[2] Sudha Bhingardive, Pushpak Bhattacharyya, “Word Sense Disambiguation Using IndoWordNet”

[3] S. Parameswarappa, V.N. Narayana, “Kannada Word Sense Disambiguation for

Machine Translation”, International Journal of Computer Applications (0975 – 8887) Volume 34– No.10, November 2011

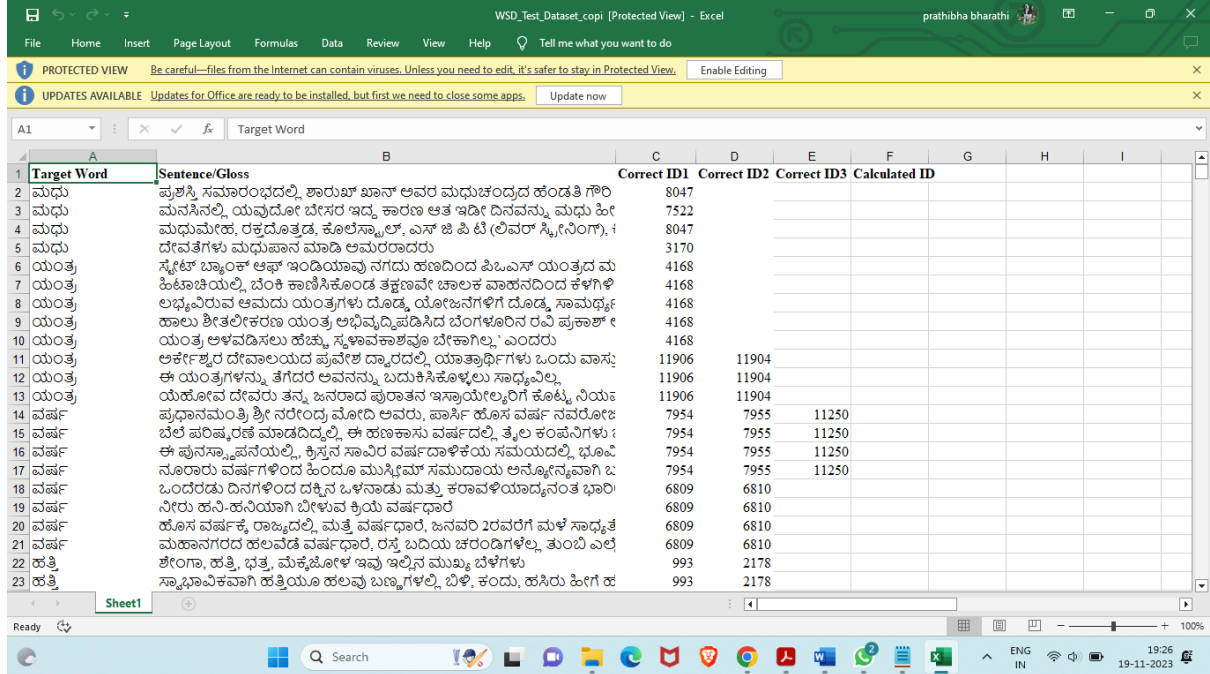
[4] [IndoWordNet - A Python based API to access Indian language WordNets](#)

I am going to use this <https://github.com/prathibhasd/pbwsd.git> for my code implementation for this report.

Test data for the Proposed system contains 100 rows.

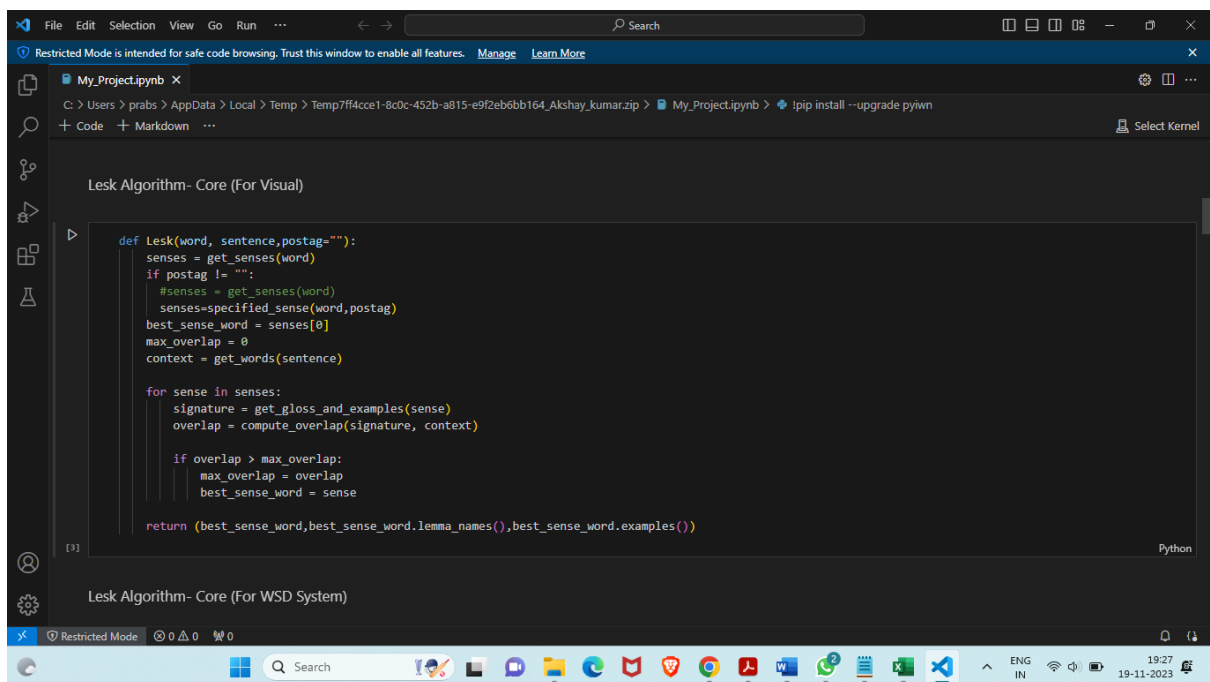
Accuracy of the System when it is tested against the test data is 54%.

Sample dataset



Target Word	Sentence/Gloss	Correct ID1	Correct ID2	Correct ID3	Calculated ID
ಮಧು	ಪ್ರಶಸ್ತಿ ಸಮಾರಂಭದಲ್ಲಿ ಶಾರುಖ್ ಖಾನ್ ಅವರ ಮಧುಚಂದ್ರದ ಹೆಂಡತಿ ಗೌರಿ	8047			
ಮಧು	ಮನಸಿನಲ್ಲಿ ಯಾವುದೋ ಬೇಸರ ಇದ್ದ ಕಾರಣ ಆತ ಇಡೀ ದಿನವನ್ನು ಮಧು ಹೀ	7522			
ಮಧು	ಮಧುಮೇಹ, ರಕ್ತದೊತ್ತಡ, ಕೊಲೆಸ್ಟ್ರಾಲ್, ಎಸ್ ಜಿ ಪಿ ಟಿ (ಲಿವರ್ ಸ್ಟ್ರೀನಿಂಗ್), ಿ	8047			
ಮಧು	ದೇವತೆಗಳು ಮಧುಪಾನ ಮಾಡಿ ಅಮರರಾದರು	3170			
ಯಂತ್ರ	ಸ್ಕೇಟ್ ಬ್ಯಾಂಕ್ ಆಫ್ ಇಂಡಿಯಾವು ನಗದು ಹಣದಿಂದ ಪಿಒಎಸ್ ಯಂತ್ರದ ಮ	4168			
ಯಂತ್ರ	ಹಿಟಾಚಿಯಲ್ಲಿ ಬೆಂಕಿ ಕಾಣಿಸಿಕೊಂಡ ತಕ್ಷಣವೇ ಚಾಲಕ ವಾಹನದಿಂದ ಕಳಗಿಳಿ	4168			
ಯಂತ್ರ	ಲಭ್ಯವಿರುವ ಅಮದು ಯಂತ್ರಗಳು ದೊಡ್ಡ ಯೋಜನೆಗಳಿಗೆ ದೊಡ್ಡ ಸಾಮರ್ಥ್ಯ	4168			
ಯಂತ್ರ	ಹಾಲು ಶೀತಲೀಕರಣ ಯಂತ್ರ ಅಭಿವೃದ್ಧಿಪಡಿಸಿದ ಬೆಂಗಳೂರಿನ ರವಿ ಪ್ರಕಾಶ್	4168			
ಯಂತ್ರ	ಯಂತ್ರ ಅಳವಡಿಸಲು ಹೆಚ್ಚು ಸ್ಥಳಾವಕಾಶವೂ ಬೇಕಾಗಿಲ್ಲ' ಎಂದರು	4168			
ಯಂತ್ರ	ಲಕ್ನೋ ಶ್ರೀ ದೇವಾಲಯದ ಪ್ರವೇಶ ದ್ವಾರದಲ್ಲಿ ಯಾತ್ರಾರ್ಥಿಗಳು ಒಂದು ವಾಸ್ತು	11906	11904		
ಯಂತ್ರ	ಈ ಯಂತ್ರಗಳನ್ನು ತೆಗೆದರೆ ಅವನನ್ನು ಬದುಕಿಸಿಕೊಳ್ಳಲು ಸಾಧ್ಯವಿಲ್ಲ	11906	11904		
ಯಂತ್ರ	ಯುರೋಪದೇವರು ತನ್ನ ಜನರಾದ ಪುರಾತನ ಇಸ್ರಾಯೇಲ್ಯರಿಗೆ ಕೊಟ್ಟ ನಿಯಮ	11906	11904		
ವರ್ಷ	ಪುರಾಣಮಂತ್ರಿ ಶ್ರೀ ನರೇಂದ್ರ ಮೋದಿ ಅವರು, ಪಾರ್ಸಿ ಹೊಸ ವರ್ಷ ನವರೋಜ	7954	7955	11250	
ವರ್ಷ	ಬೆಲೆ ಪರಿಷ್ಕರಣೆ ಮಾಡದಿದ್ದಲ್ಲಿ ಈ ಹಣಕಾಸು ವರ್ಷದಲ್ಲಿ ತೈಲ ಕಂಪನಿಗಳು	7954	7955	11250	
ವರ್ಷ	ಈ ಪುನಸ್ಸಾಂಪನೆಯಲ್ಲಿ ಕ್ರಿಸ್ತನ ಸಾವಿರ ವರ್ಷದಾಳಿಕೆಯ ಸಮಯದಲ್ಲಿ ಭೂವಿ	7954	7955	11250	
ವರ್ಷ	ನೂರಾರು ವರ್ಷಗಳಿಂದ ಹಿಂದೂ ಮುಸ್ಲಿಮ್ ಸಮುದಾಯ ಅನ್ಯೋನ್ಯವಾಗಿ ಒ	7954	7955	11250	
ವರ್ಷ	ಒಂದೆರಡು ದಿನಗಳಿಂದ ದಕ್ಕಿನ ಒಳನಾಡು ಮತ್ತು ಕರಾವಳಿಯಾದ್ಯಂತ ಭಾರಿ	6809	6810		
ವರ್ಷ	ನೀರು ಹಸಿ-ಹಸಿಯಾಗಿ ಬೀಳುವ ಕ್ರಿಯೆ ವರ್ಷಧಾರೆ	6809	6810		
ವರ್ಷ	ಹೊಸ ವರ್ಷಕ್ಕೆ ರಾಜ್ಯದಲ್ಲಿ ಮತ್ತೆ ವರ್ಷಧಾರೆ, ಜನವರಿ 2ರವರೆಗೆ ಮಳೆ ಸಾಧ್ಯತೆ	6809	6810		
ವರ್ಷ	ಮಹಾನಗರದ ಹಲವೆಡೆ ವರ್ಷಧಾರೆ, ರಸ್ತೆ ಬದಿಯ ಚರಂಡಿಗಳಲ್ಲಿ ತುಂಬಿ ಎಲ್ಲೆ	6809	6810		
ಹತ್ತಿ	ಶೌಂಗಾ, ಹತ್ತಿ, ಭತ್ತೆ, ಮೆಕ್ಕೆಜೋಳ ಇವು ಇಲ್ಲಿನ ಮುಖ್ಯ ಬೆಳೆಗಳು	993	2178		
ಹತ್ತಿ	ಸಾಾಭಾವಿಕವಾಗಿ ಹತ್ತಿಯೂ ಹಲವು ಬಣ್ಣಗಳಲ್ಲಿ ಬಿಳಿ, ಕಂದು, ಹಸಿರು ಹೀಗೆ ಹ	993	2178		

Lesk algorithm visual



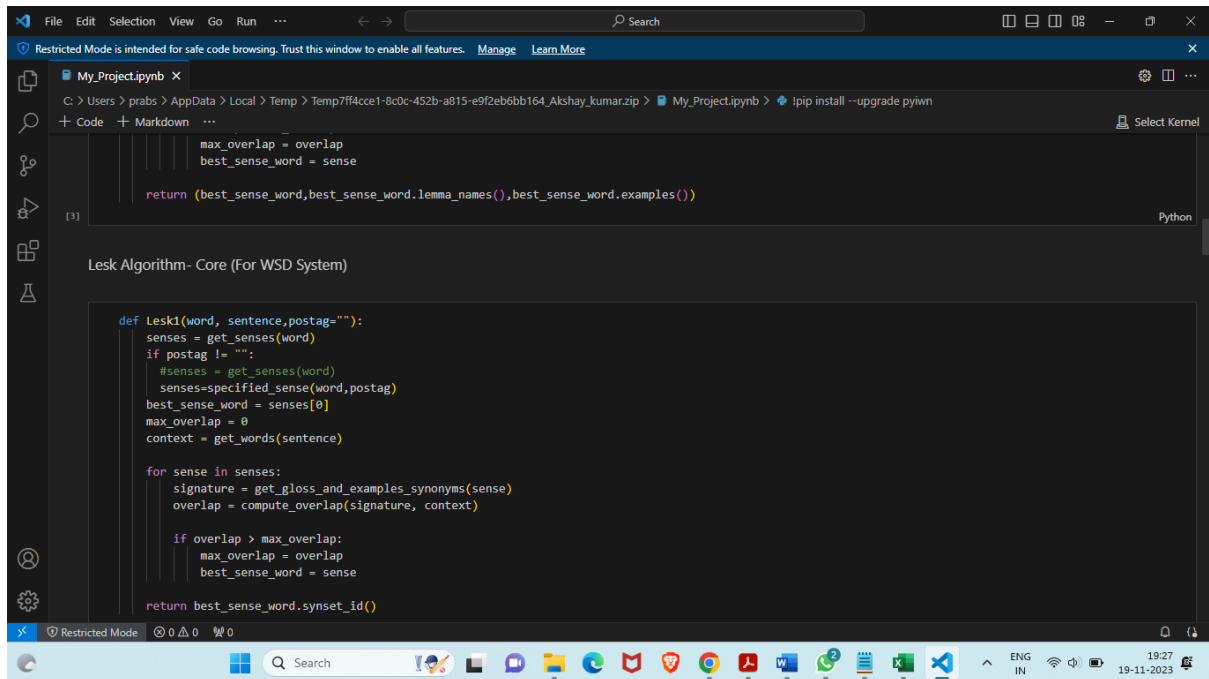
```
def Lesk(word, sentence, postag=""):
    senses = get_senses(word)
    if postag != "":
        #senses = get_senses(word)
        senses=senses+specified_sense(word,postag)
    best_sense_word = senses[0]
    max_overlap = 0
    context = get_words(sentence)

    for sense in senses:
        signature = get_gloss_and_examples(sense)
        overlap = compute_overlap(signature, context)

        if overlap > max_overlap:
            max_overlap = overlap
            best_sense_word = sense

    return (best_sense_word,best_sense_word.lemma_names(),best_sense_word.examples())
```

Lesk algorithm for word sense Disambiguation



The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes a search bar and a 'Restricted Mode' warning. The notebook is titled 'My_ProjectIpyb'. The code is written in Python and implements the Lesk algorithm for word sense disambiguation. The code defines a function 'Lesk1' that takes a word, a sentence, and a postag as input. It uses NLTK's 'get_senses' function to retrieve senses for the word. The function then iterates through these senses, calculating the overlap between the sense's gloss and the words in the sentence. The sense with the highest overlap is selected as the best sense. The function returns the best sense word, its lemma names, and its examples.

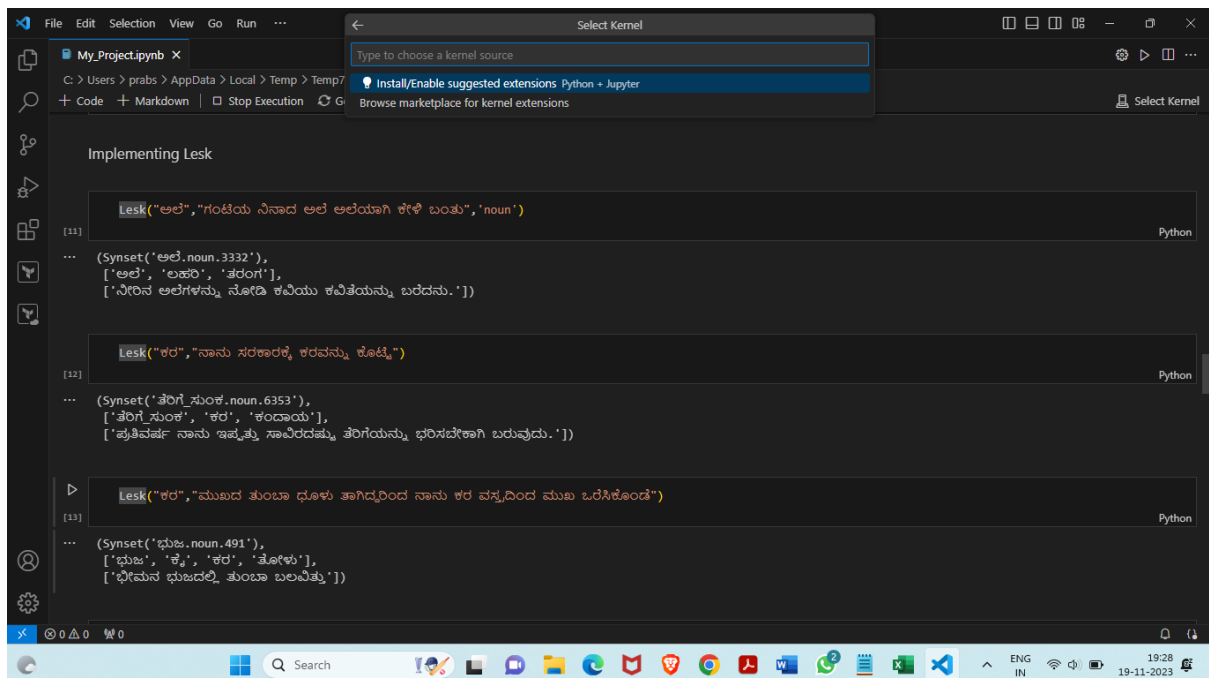
```
def Lesk1(word, sentence, postag=""):
    senses = get_senses(word)
    if postag != "":
        #senses = get_senses(word)
        senses=senses_specified_sense(word, postag)
    best_sense_word = senses[0]
    max_overlap = 0
    context = get_words(sentence)

    for sense in senses:
        signature = get_gloss_and_examples_synonyms(sense)
        overlap = compute_overlap(signature, context)

        if overlap > max_overlap:
            max_overlap = overlap
            best_sense_word = sense

    return best_sense_word.synset_id()
```

Implementing lesk on dataset



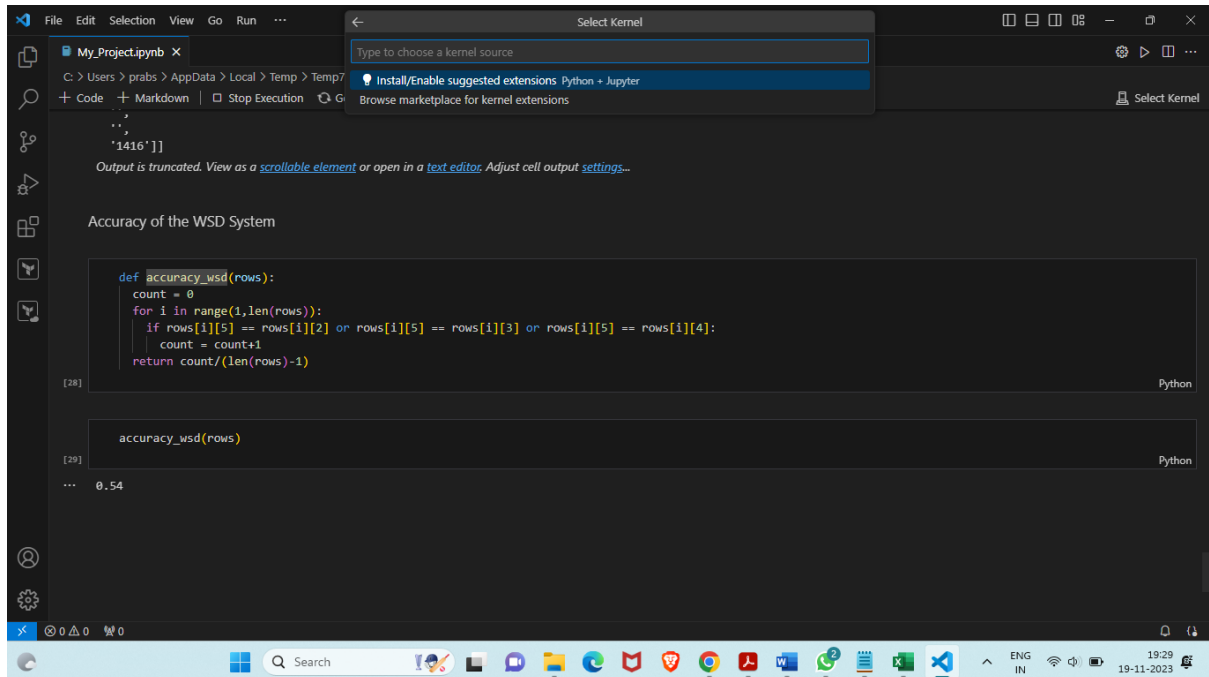
The screenshot shows a Jupyter Notebook interface with a dark theme. The top bar includes a search bar and a 'Restricted Mode' warning. The notebook is titled 'My_ProjectIpyb'. The code is written in Python and implements the Lesk algorithm on a dataset. The code defines a function 'Lesk' that takes a word and a sentence as input. It uses NLTK's 'get_senses' function to retrieve senses for the word. The function then iterates through these senses, calculating the overlap between the sense's gloss and the words in the sentence. The sense with the highest overlap is selected as the best sense. The function returns the best sense word, its lemma names, and its examples.

```
Lesk("ಅಲೆ", "ಗಿಂಟಿಯ ನಿನ್ನಾದ ಅಲೆ ಅಲೆಯಾಗಿ ಕೇಳಿ ಬಂತು", "noun")
(Synset('ಅಲೆ.noun.3332'),
 ['ಅಲೆ', 'ಲಪರಿ', 'ಪರಂಗಿ'],
 ['ನಿರೀನ ಅಲೆಗಳನ್ನು ನೋಡಿ ಕವಿಯು ಕವಿತೆಯನ್ನು ಬರೆದನು.'])

Lesk("ಕರ", "ನಾನು ಸರಕಾರಕ್ಕೆ ಕರವನ್ನು ಕೊಟ್ಟೆ")
(Synset('ಕರಿಗೆ ಸಂಕ. noun.6353'),
 ['ಕರಿಗೆ ಸಂಕ', 'ಕರ', 'ಕಂದಾಯ'],
 ['ಪ್ರತಿವರ್ಷ ನಾನು ಇಪ್ಪತ್ತು ಸಾವಿರದಷ್ಟು ಕರಿಗಳನ್ನು ಭರಿಸಬೇಕಾಗಿ ಬರುವುದು.'])

Lesk("ಕರ", "ಮುಖದ ತುಂಬಾ ಧೂಳು ತಾಗಿದ್ದರಿಂದ ನಾನು ಕರ ವಸ್ತ್ರದಿಂದ ಮುಖ ಒರೆಸಿಕೊಂಡೆ")
(Synset('ಭುಜ. noun.491'),
 ['ಭುಜ', 'ಕೊ', 'ಕರ', 'ತೋಳು'],
 ['ಭೀಮನ ಭುಜದಲ್ಲಿ ತುಂಬಾ ಬಲವಿತ್ತು.'])
```

Accuracy:



The screenshot shows a Jupyter Notebook window titled 'My_Project.ipynb'. The notebook is open to a cell containing a Python function named `accuracy_wsd`. The function takes a list of rows as input and returns a count of rows where the word at index 5 is equal to the word at index 2, 3, or 4. The function is called with the input `rows`, and the output is `0.54`. The notebook interface includes a 'Select Kernel' dialog box at the top, which is currently set to 'Python + Jupyter'. The notebook also shows a 'Code' tab and a 'Markdown' tab. The output of the cell is truncated, showing only the first few lines of the function definition and the result of the function call.

```
def accuracy_wsd(rows):
    count = 0
    for i in range(1, len(rows)):
        if rows[i][5] == rows[i][2] or rows[i][5] == rows[i][3] or rows[i][5] == rows[i][4]:
            count = count + 1
    return count / (len(rows) - 1)

accuracy_wsd(rows)

... 0.54
```

Business Model (Monetization Idea)

Consider enhancing contextual advertising platforms with your WSD technology. Many online advertisers struggle with ensuring that their ads are displayed in contexts that match the intended meaning of their brand or product. By integrating your WSD solution into advertising platforms, you can offer advertisers the ability to target their ads more accurately based on the contextual meaning of the content.

Offer a service where you analyse and optimize the contextual relevance of specific advertising campaigns using your WSD technology. Charge advertisers a fee for the optimization service, positioning it as a value-added solution to improve ad performance.

Integrate WSD into the chatbot's natural language processing pipeline to better understand the context and intended meaning of user input. Ensure that the chatbot can accurately identify and disambiguate words with multiple meanings, providing a more nuanced comprehension of user queries.

Concept Generation (process of coming up with Idea)

The proposed approach is implemented using Indo-WordNet as the primary lexical resource. For experiments, we used the dataset given in Table 1 and input sentences are taken from the ‘Samananthara dataset’, Kannada blogs, e-newspapers, Kannada articles etc.

Concept development:

Implementation of the proposed algorithm is illustrated with the following example.

Consider the following Table: Target word contains the Word for disambiguation for the corresponding sentence. IndoWordNet gives the Example sentence and Gloss for the Target word.

Final Product prototype:

The Architecture

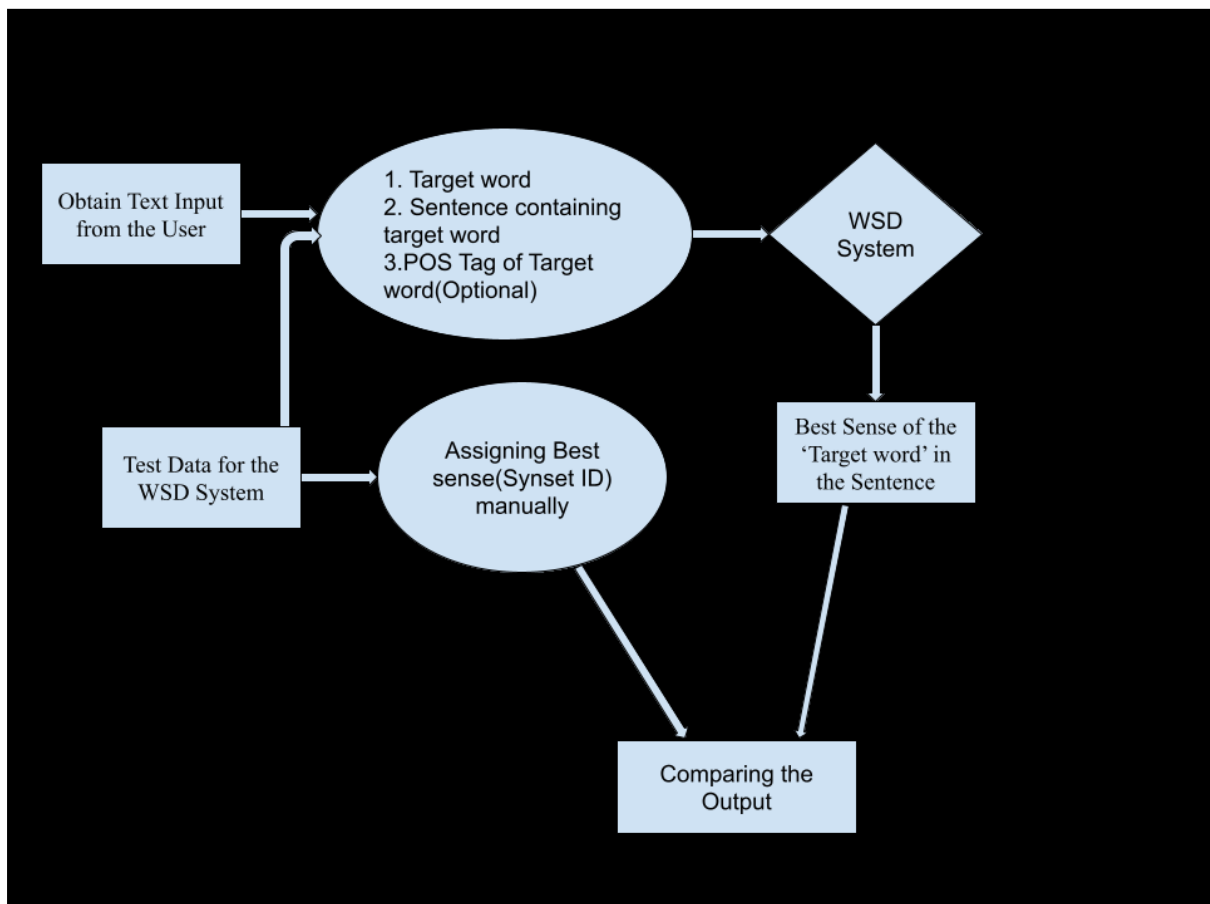


Fig 1. Project architecture

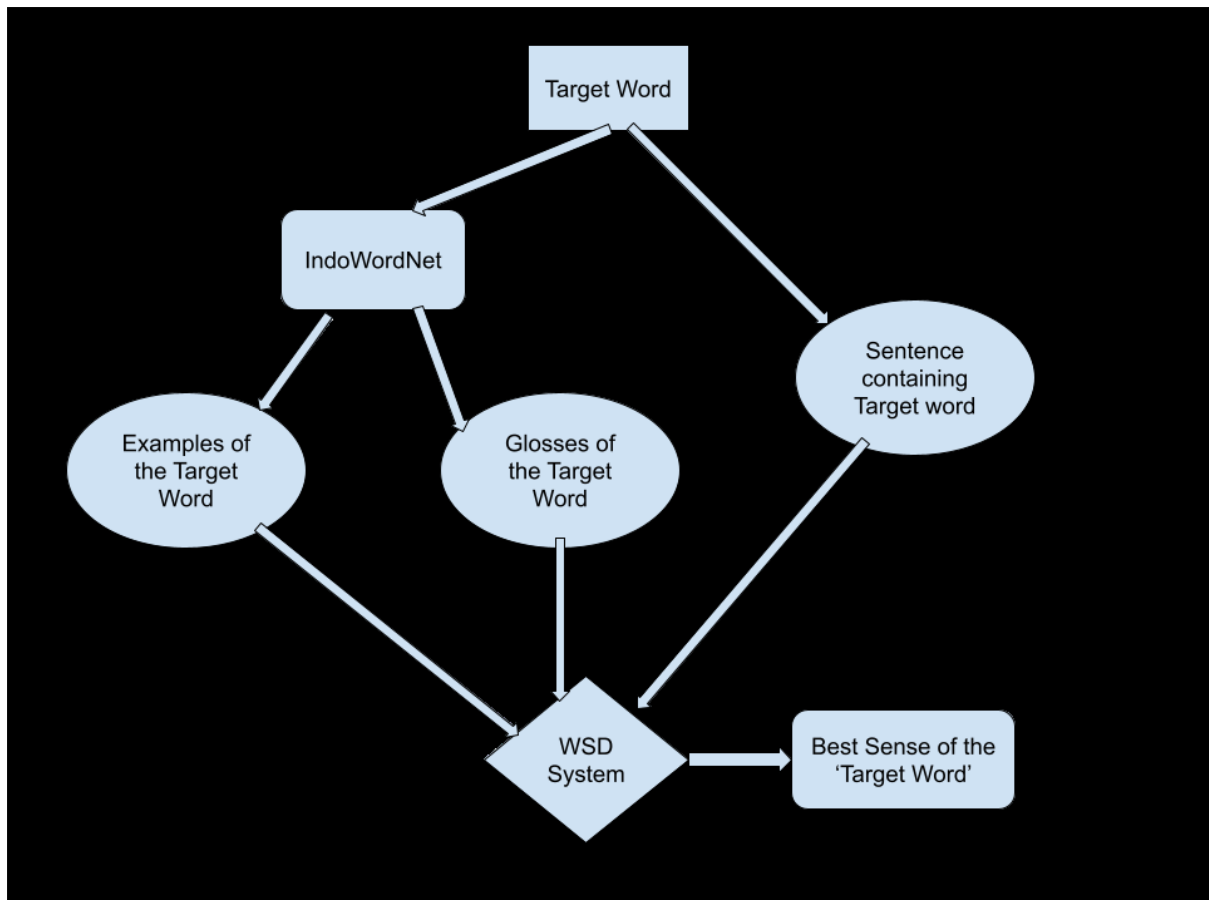


Fig 2. Wsd system

Lexical Methods:

- Use pre-existing sense inventories from lexical resources.
- Analyse word features, such as definitions, example sentences, and part-of-speech information.
- Compute similarity measures between word features and context.

Integrate with Chatbot:

- Context Analysis:
 - Extract context from the ongoing conversation.
 - Define the scope of context, such as the current message, previous messages, or a specific context window.
- Identify Ambiguous Words:
 - Identify words with multiple senses that require disambiguation.

Feasibility

Scope and Objectives:

Clearly define the scope and objectives of your WSD project. Understand the specific use case or application where word sense disambiguation is needed.

Data Availability:

Assess the availability and quality of the data required for training and evaluating the WSD model. Adequate and representative data is crucial for the success of a machine learning-based WSD system.

Algorithmic Approaches:

Explore different algorithmic approaches for WSD. Consider whether rule-based, supervised learning, unsupervised learning, or a combination of methods is most suitable for your project.

Integration with Other Systems:

Consider how the WSD system will integrate with other components of your project or existing systems. Compatibility and seamless integration are essential for practical applications.

Viability

Accuracy:

The primary factor in evaluating WSD software is its accuracy in determining the correct sense of words within a given context. High accuracy is essential for the software to be reliable and valuable.

Context Understanding:

The software should demonstrate a robust understanding of contextual nuances to accurately disambiguate word senses. It should be able to consider surrounding words, phrases, and broader linguistic context.

Data Quality and Availability:

The quality and availability of training data significantly impact the viability of WSD software. A diverse and representative dataset is crucial for training a model that can generalize well to various contexts.

Financial Modelling

It can be directly launched as chatbot or software to the market.

Creating a financial equation for word sense disambiguation (WSD) involves considering the costs associated with developing and implementing a WSD system. While the financial aspects can be complex and vary depending on the specific project and organization, here is a simplified representation:

Total Cost=Development Cost + Operational Cost

Development Cost:

This includes expenses related to building the WSD system. It encompasses costs associated with: Data acquisition and preprocessing: Obtaining and preparing labelled datasets for training.

Algorithm development: Designing and implementing the WSD algorithm.

Model training: Using computational resources to train the WSD model.

Software development: Creating the actual WSD software or integrating it into existing systems.

Testing and validation: Ensuring the accuracy and effectiveness of the WSD system.

Operational Cost:

This includes ongoing expenses associated with running and maintaining the WSD system after development. It encompasses costs such as:

Hosting and infrastructure: Expenses related to server hosting and computational resources for deploying the WSD system.

Maintenance and updates: Costs associated with regular updates, bug fixes, and improvements to the WSD software.

Monitoring and support: Resources allocated to monitoring system performance and providing support to end-users.

Training and retraining: Expenses related to keeping the WSD model up-to-date with new data or adapting it to changes in language usage.

CONCLUSION

In this project, we have highlighted the role of IndoWordNet for performing Word Sense Disambiguation for Indian languages. IndoWordNet is used as a sense repository which consists of unique concepts, its semantic relations, lexical relations between words, etc. We have presented an approach for WSD by implementing the Lesk algorithm which uses the IndoWordNet sense repository. Development of a large WordNet base for Kannada language is essential to store all types of words of this vast, old, historically rich language. All the chunks of synonyms, antonyms, homonyms, hyponyms are to be collected and updated in the WordNet database. Along with the WordNet, there is also an equal opportunity for the creation of Kannada corpus consisting of words from various resources such as novels, biographies, magazines, books, etc. This project is only a stepping stone towards natural language processing for Kannada language in machine learning. With this development, the word sense disambiguation will be much more accurate and more efficient.