Word Sense Disambiguation: PageRank and Lesk Algorithm

CHAYAPATR ARCHIWARANGUPROK

This paper presents a combined approach to Word Sense Disambiguation (WSD) by combining the structural analysis capabilities of the PageRank algorithm with the contextual understanding of the Lesk algorithm. Our hybrid method creates a sense graph representing semantic relationships between word senses and utilizes both graph centrality and definitional overlap to determine the most appropriate word sense in a given context. The approach also incorporates multi-word expression (MWE) detection to improve disambiguation accuracy.

1 INTRODUCTION

Word Sense Disambiguation (WSD) is one of the most fundamental challenges in Natural Language Processing (NLP): determining which meaning of a word is being used in a given context [5]. Consider the word "bank" - it could refer to a financial institution, the edge of a river, or the act of tilting an aircraft. Humans typically resolve such ambiguities effortlessly through context, but enabling computers to do the same has proven to be a persistent challenge in NLP.

The ability to accurately disambiguate word senses is crucial for many downstream NLP tasks. Machine translation systems need to know whether to translate "bank" to a financial term or a geographical feature. Information retrieval systems need to understand whether a user searching for "java" is interested in coffee, the programming language, or the island. Similarly, text-to-speech systems might need to distinguish between "lead" (the metal) and "lead" (to guide) to produce correct pronunciations.

In this paper, we experiment with combining two methods for a simple yet powerful WSD, Knowledge-based and Graph-based method, to create a more robust disambiguator. Knowledge-based methods, such as the Lesk algorithm, compare the dictionary definitions of possible word senses with the context in which the word appears. Graph-based methods, like PageRank, treat word senses as nodes in a semantic network and use graph algorithms to identify the most likely sense based on relationships between words. Each approach has its strengths: knowledge-based methods can leverage rich lexical resources, while graph-based methods can capture broader semantic relationships.

2 METHODOLOGY

Our approach consists of several key components that work together to perform WSD. This section provides detail each component and its role in the overall system. In this system, we input a sentence for the WSD to perform disambiguation and provide meaning for each word.

2.1 Sense Graph Creation

First, we construct a sense graph that captures the semantic relationships between word senses in input sentence. The graph construction process transforms raw text into a weighted undirected graph $G = (V, E, W)$, where vertices V represent word senses (synsets), edges E represent semantic relationships, and weights W represent the strength of these relationships.

2.1.1 Text Preprocessing. Before graph creation, we run initial preprocessing to transform raw text into a format suitable for analysis through several steps:

(1) **Case normalization**: Convert all text to lowercase to ensure consistent processing
(2) **Tokenization**: Split text into individual tokens using NLTK's word tokenizer
(3) **Stopword removal**: Filter out common words (e.g., "the", "is", "at") that typically don't carry sense ambiguity

Author's address: Chayapatr Archiwaranguprok, pub@from.pub.

(4) **Punctuation removal**: Strip punctuation marks that don't contribute to word meaning

Since each word can serve multiple parts of speech, we narrow down the word senses to only those that match the original meaning. We perform part-of-speech tagging using NLTK's tagger [1], which assigns Penn Treebank tags to each token. These tags are then mapped to WordNet's simplified POS categories.

2.1.2 Multi-word Expression Identification. Multi-word expressions (MWEs) present a unique challenge in WSD, as their meaning often cannot be derived from the individual meanings of their constituent words. Our system implements a compound word detection mechanism that:

- Examines adjacent word pairs in the input text
- Queries WordNet for compound form existence
- Maintains original POS tagging information

This process is formalized as:

$$MWE(w_i, w_{i+1}) = \begin{cases} (w_i\_w_{i+1}, pos_i) & \text{if } \exists \text{ synsets}(w_i\_w_{i+1}) \\ (w_i, pos_i) & \text{otherwise} \end{cases}$$

2.1.3 Graph Construction. The graph construction process operates within a sliding context window of size $2k + 1$ centered on each target word, where k is the window size (we use k=5 as a default). For each word w_i at position i in the text:

- **Vertex Creation**: For each word with a valid POS tag, we retrieve all possible WordNet [4] synsets:
$$V_i = s \mid s \in \text{synsets}(w_i, pos(w_i))$$
- **Edge Creation**: Within the context window $[i - k, i + k]$, we create edges between all pairs of synsets from different words:
$$E_{ij} = (s_1, s_2) \mid s_1 \in V_i, s_2 \in V_j, i \neq j$$
- **Weight Assignment**: Each edge is weighted using WordNet's path similarity measure, which quantifies the semantic relatedness of two synsets based on their positions in WordNet's hierarchical structure. Since WordNet is organized as a taxonomic tree, particularly for nouns and verbs, we can calculate the shortest path length between two synsets by traversing the hypernym/hyponym relationships:
$$w(s_1, s_2) = \text{path\_similarity}(s_1, s_2)$$

The final graph G captures both the strong local context, achieved through window-based edge creation, and the weaker global semantics, with each word sense represented as a vertex. This structure serves as input to the PageRank algorithm, which utilizes both random walks and warping to determine the most likely sense for each word.

2.2 PageRank Algorithm

The PageRank [6] is one of the two parts in our combined WSD approach. The algorithm analyzes the global structure of the sense graph to determine the relative importance of each word sense. While traditional PageRank was developed for web page ranking, we adapt it to work with weighted semantic graphs where vertices represent word senses and edges represent semantic similarities.

For a given sense graph $G = (V, E, W)$, The algorithm begins by initializing all vertices with an equal PageRank score:

$$PR^0(v_i) = \frac{1}{|V|} \quad \forall v_i \in V$$

.

the PageRank score for each vertex v_i is then computed through an iterative process until convergence:

$$PR^{t+1}(v_i) = (1-d) + d \sum_{v_j \in N(v_i)} \frac{PR^t(v_j)}{|N(v_j)|} \cdot w_{ji}$$

where:

- $PR^t(v_i)$ is the PageRank score for vertex v_i at iteration t
- d is the damping factor (we set the value to 0.85)
- $N(v_i)$ represents the set of vertices adjacent to v_i
- $|N(v_j)|$ is the number of outgoing edges from vertex v_j
- w_{ji} is the semantic similarity weight between vertices v_j and v_i

At each iteration, vertices receive weighted contributions from their neighbors based on edge weights and current PageRank scores. A damping factor d introduces random jumps with probability $(1-d)$, transforming the adjacency matrix into a primitive stochastic matrix (irreducible and aperiodic), ensuring convergence for the underlying Markov Chain [2] to a unique solution. The final PageRank scores $PR(v_i)$ represent each word sense's global importance, integrating both:

(1) Direct relationships captured through walks between adjacent vertices in the graph (length-1 paths), representing immediate semantic connections between word senses
(2) Indirect relationships captured through warps (paths of length > 1), which reveal semantic connections mediated through intermediate word senses

These scores are then combined with local context information from the Lesk algorithm during the disambiguation phase, as detailed in the following parts.

2.3 Lesk Algorithm

The Lesk algorithm [3] provides complementary information by analyzing the lexical overlap between sense definitions and the target word's context. For each candidate sense, it measures how many words are shared between the sense's dictionary entry and the context in which the ambiguous word appears. Our implementation computes a Lesk score:

$$\text{Lesk}(s, c) = |\text{signature}(s) \cap \text{context}(c)|$$

where:

- s is a candidate sense from WordNet
- c is the context (the preprocessed sentence containing the target word)
- $\text{signature}(s)$ is the bag of words from both the sense's definition and its usage examples
- $\text{context}(c)$ is the bag of words from the input sentence

The signature of a sense is constructed by combining two sources of information from WordNet:

(1) The sense's definition (gloss), which provides a formal explanation of the meaning
(2) Usage examples, which demonstrate the sense in context

Both the signature and context undergo preprocessing steps such as tokenization, stopword removal, and lemmatization to normalize the text before comparison. This ensures that morphological variants of the same word (e.g., "run", "runs", "running") are matched correctly. The final score is simply the size of the intersection between these two preprocessed sets of words, with higher

scores indicating stronger evidence for that particular sense. For example, given the word "bank" in the sentence "I need to bank the airplane for landing", the algorithm would compare this context against each sense's signature. The aviation-related sense of "bank" would likely score higher as its definition and examples contain words that overlap with the context ("airplane", "landing").

For practical implementation, our function LeskSim($word, sentence, pos$) computes Lesk scores for all possible synsets of the target word. Given a word, its part of speech, and the containing sentence, it retrieves all candidate synsets from WordNet and returns a dictionary mapping each synset to its Lesk score. For each synset, it builds the signature by combining preprocessed words from both the definition and examples, then computes the intersection size with the preprocessed context words.

2.4 Disambiguator

The final disambiguation step integrates the PageRank and Lesk scores through a carefully designed algorithm that processes each word in context. The process is formally defined in Algorithm 1.

Algorithm 1 Word Sense Disambiguation with PageRank and Lesk Algorithm

function DISAMBIGUATE(G, tagged_words, sentence)
  $results \leftarrow \emptyset$
  $confidence\_scores \leftarrow \emptyset$
  $pagerank\_scores \leftarrow PageRank(G)$ ▷ Calculate PageRank
  for $(word, tag)$ in $tagged\_words$ **do**
    $pos \leftarrow GetWordNetPOS(tag)$
    if pos is valid **then**
      $synsets \leftarrow WordNet.getSynsets(word, pos)$
      if $synsets$ not empty **then**
        $lesk\_scores \leftarrow LeskSim(word, sentence, pos)$ ▷ Calculate Lesk Similarity
        $common\_senses \leftarrow Keys(pagerank\_scores) \cap Keys(lesk\_scores)$
        $scores \leftarrow \{s : pagerank\_scores[s] \times lesk\_scores[s] \mid s \in common\_senses\}$
        $best\_sense \leftarrow \arg\max(scores)$
        $results[word] \leftarrow best\_sense$
        $confidence\_scores[word] \leftarrow scores[best\_sense]$
      end if
    end if
  end forreturn $(results, confidence\_scores)$
end function

In short, the algorithm combines the score from PageRank ($PR(s)$) with contextual information from Lesk ($Lesk(s, c)$) through multiplication:

$$score(s) = PR(s) \cdot Lesk(s, c)$$

This multiplicative combination ensures that a sense must score reasonably well in both metrics to be selected, as a poor score in either metric will significantly reduce the combined score. The best sense for each word is then selected by maximizing this combined score:

$$best\_sense(w) = \arg\max_{s \in senses} score(s)$$

The algorithm returns both the selected senses and their confidence scores, allowing downstream applications to filter or threshold results based on confidence levels. This integration of global

graph structure with local contextual similarity provides a robust framework for word sense disambiguation.

3 RESULT AND DISCUSSION

We demonstrate our system's disambiguation capabilities with two examples using the word "bank" in different contexts:

"I was depositing money at the bank this morning."

Disambiguated as:

- **bank**: a financial institution that accepts deposits and channels the money into lending activities (Confidence: 0.08)
- **money**: the official currency issued by a government or national bank (Confidence: 0.12)

"The nearby river bank was flooded yesterday."

Disambiguated as:

- **bank**: sloping land (especially the slope beside a body of water) (Confidence: 0.09)
- **river**: a large natural stream of water (larger than a creek) (Confidence: 0.08)

In this case, the system correctly disambiguates "bank" as the geographical feature, influenced by the presence of "river" and "flooded" in the context. These examples demonstrate the system's ability to leverage contextual cues to select appropriate word senses. The confidence scores, while seemingly low due to the inherent complexity of the disambiguation task, show relative certainty differences between competing senses. Notably, content words (like "money" and "flooded") tend to receive higher confidence scores than function words or temporal expressions.

3.1 Future Work

Several potential improvements could enhance the system's performance. One key area is handling sentences that contain multiple instances of the same ambiguous word. Consider the sentence "The bank teller processed my deposit while I watched boats pass by the river bank" – our current approach might struggle to correctly disambiguate both instances of "bank" simultaneously. Incorporating syntactic parsing to create separate subgraphs for different clauses could help isolate and better disambiguate these cases.

Additionally, the similarity measures could be enhanced by incorporating modern contextual embeddings. While our current approach relies on WordNet's taxonomy for measuring semantic relationships, pretrained language models could provide more nuanced similarity scores that capture broader contextual patterns learned from large-scale corpora. For instance, BERT embeddings could help identify subtle relationships between words that might not be explicitly encoded in WordNet's hierarchy.

3.2 Remark

The code for the paper is available at https://colab.research.google.com/drive/1y1oLrWwTdA_PjDzGL1L8SpL9IKGkFsFc?usp=sharing.

REFERENCES

[1] Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.".

[2] Jeff Jauregui. 2012. Markov chains, Google's PageRank algorithm. https://www2.math.upenn.edu/~kazdan/312F12/JJ/MarkovChains/markov_google.pdf. [Accessed 28-11-2024].

[3] Michael Lesk. 1986. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation* (Toronto,

Ontario, Canada) *(SIGDOC '86)*. Association for Computing Machinery, New York, NY, USA, 24–26. https://doi.org/10.1145/318723.318728

[4] George A. Miller. 1995. WordNet: a lexical database for English. *Commun. ACM* 38, 11 (Nov. 1995), 39–41. https://doi.org/10.1145/219717.219748

[5] Roberto Navigli. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41, 2, Article 10 (Feb. 2009), 69 pages. https://doi.org/10.1145/1459352.1459355

[6] Lawrence Page. 1999. *The PageRank citation ranking: Bringing order to the web.* Technical Report. Technical Report.