TLN Part 2 - Word Sense Disambiguation

Paletto Andrea, Tuninetti André

September 5, 2023

Contents

1 Introduction

In this task, we will implement the Simplified Lesk algorithm from scratch to perform word sense disambiguation. The goal is to disambiguate the senses of nouns in sentences using the context provided by the SemCor corpus. This task involves the following steps:

1. **Data Preparation**: We will obtain the SemCor corpus, which is annotated with WordNet synsets. The SemCor corpus is available for download from the following URL: SemCor Corpus.

2. **Implementation and Evaluation**: We will extract 50 sentences from the SemCor corpus and perform word sense disambiguation for at least one noun in each sentence. The Lesk algorithm will be implemented from scratch for this purpose. We will then calculate the accuracy of our system by comparing the assigned senses with the annotated senses in SemCor.

3. **Randomization and Average Accuracy**: To ensure robustness, we will randomly select 50 sentences from the SemCor corpus and perform word sense disambiguation for randomly chosen nouns. This process will be repeated multiple times (e.g., 10 times), and the average accuracy of our system's performance across these executions will be calculated.

4. **Corpus-Lesk Algorithm**: An implementation of the Lesk algorithm using the SemCor corpus will be provided.

2 Methodology

2.1 Overview

This section describes the methodology and functions used in the implementation of a simplified version of the Lesk algorithm for word sense disambiguation using WordNet synsets. The Lesk algorithm aims to determine the most suitable sense of a word within a given context by comparing the overlap of the word's signature with the context.

2.2 Functions

- `remove_punctuation(sentence)` This function removes punctuation from a sentence, excluding hyphens and underscores.

- `signature(sense, stop_words)` This function computes the signature of a WordNet synset. The signature includes the synset's definition and words from its examples, excluding specified stop words.

- `compute_overlap(signature, context)` This function calculates the overlap between a synset's signature and a given context in terms of the number of shared words.

- `extract_nouns_index(sentence, pos)` This function identifies the indexes of nouns in a sentence based on their POS tags.
- `simplified_lesk(word, sentence, stop_words)` This function performs a simplified Lesk algorithm. It finds the synset that best matches the given word within a specific sentence context, using the overlap between synset signatures and context words.
- `semcor_extraction(sentence_number=50)` This function extracts a specified number of sentences containing random nouns from the SemCor corpus. It returns the extracted sentences along with the extracted noun lemmas.
- `select_lemma(lemmas)` This function randomly selects a lemma from a list of lemmas, ensuring it represents a single word.
- `remove_word(sentence, word)` This function removes a chosen word from a sentence.

2.3 Workflow

The preprocessing follows these steps:

1. Import necessary libraries and modules.
2. Define various utility functions for text processing and Lesk algorithm steps.
3. Specify the path to the SemCor corpus.
4. Load stop words from a file.
5. Define the `semcor_extraction()` function to extract random sentences and their associated noun lemmas from SemCor.
6. Provide the `select_lemma()` function to select a lemma representing a single word.
7. Implement the `remove_word()` function to remove a chosen word from a sentence.

The **evaluation process** involves the following steps:

1. **Accuracies List Initialization**
   An empty list called `accuracies` is created to store the accuracy of each run.

2. **Loop for Multiple Runs**
   The code enters a loop that runs `num_runs` times to evaluate accuracy over multiple iterations.

3. **Disambiguation Loop**
   Inside the loop:

   - Sentences and corresponding nouns are extracted from SemCor using `semcor_extraction`.
   - An empty list named `results` is created to hold disambiguation results.

4. **Disambiguating and Comparing**
   For each sentence-noun pair:

   - The `simplified_lesk` function is used to perform sense disambiguation on the noun in the sentence.
   - The disambiguated sense is compared with the sense annotated in SemCor for evaluation.

5. **Recording and Calculating Results**
   If the disambiguated sense matches the SemCor sense, 1 is added to `results`; otherwise, 0 is added. Accuracy for the current run is calculated as the sum of correct disambiguations divided by the total number of pairs.

6. **Accuracies Collection**
   The calculated accuracy for the current run is added to the `accuracies` list.

7. **Calculating Average Accuracy**
   After all runs, the code calculates the average accuracy across all runs using the `np.mean` function.

8. **Printing Results**
   The average accuracy over `num_runs` runs is printed to the console.

This evaluation process assesses the performance of the Lesk algorithm for word sense disambiguation by running multiple iterations and calculating the average accuracy.

3 Conclusions

3.1 Evaluation Results

After running the Lesk algorithm for word sense disambiguation using your provided code, here are some key observations based on the output:

- The Lesk algorithm attempted to disambiguate word senses for a set of sentences extracted from the SemCor corpus.
- For each sentence, the algorithm assigned a sense using the `simplified_lesk` function and compared it to the annotated sense from SemCor.
- The output shows instances where the disambiguated sense matched the annotated sense and instances where it did not.

3.2 Accuracy Analysis

The provided output includes disambiguation results for different sentences, along with the senses assigned by the Lesk algorithm and the annotated senses from SemCor. The calculated average accuracy over 10 runs is approximately 0.45 (45%). This average accuracy indicates that the Lesk algorithm's performance is moderate, suggesting that it correctly identified the sense of the target words in about 45% of the cases.

3.3 Factors Impacting Performance

Several factors could contribute to the observed accuracy:

- **Sense Overlap**: The algorithm's accuracy heavily depends on the overlap between the context words and the words in the synset signature. If there's limited overlap, the algorithm may struggle to identify the correct sense.
- **Context Complexity**: Complex sentence structures or sentences with multiple possible interpretations can challenge the algorithm's accuracy.
- **Quality of Annotations**: The accuracy also depends on the quality of annotations in the SemCor corpus. If the corpus contains ambiguous or incorrect annotations, it can affect the evaluation.