TLN Part 3 Lab 01 - Defs

Paletto Andrea, Tuninetti André

September 12, 2023

Contents

1 Introduction

In this exercise we implemented an experiment named "Defs" to calculate similarity and perform aggregation on dimensions of concreteness and specificity using word definitions. The experiment involves processing definitions of these four words: "door", "ladybug", "pain", and "blurriness". The primary objective is to compute similarity scores between pairs of definitions and subsequently aggregate the scores based on the dimensions of concreteness and specificity.

2 Methodology & Workflow

2.1 Importing Libraries and Loading Data

We start by importing essential libraries, such as `nltk`, `spacy`, and `csv`. Then we load definitions from a TSV file into a dictionary named `definitions`, where the keys represent words, and the values are lists of corresponding definitions.

2.2 Defining Functions

1. `compute_similarity(definition1, definition2)`: This function calculates the similarity between two definitions. It preprocesses the text of both definitions, finds the intersection of words, and computes the normalized intersection size over the minimum length of the two definitions.

2. `words_frequency(definition1, definition2)`: This function calculates the similarity between two definitions based on the frequency of words in the definitions. It calculates the normalized word frequency for the most frequent word that appears in both definitions. The similarity score is determined by the frequency of the most common word that is common to both definitions relative to the total number of words in both definitions. This approach gives higher scores to definitions where the most frequent word is common between them.

3. `preprocess_text(text)`: This function tokenizes and preprocesses the given text. It converts the text to lowercase, tokenizes it, removes stop words, and performs stemming using the Porter Stemmer.

2.3 Similarity and Aggregation

For each word the code performs the following steps:

1. Calculates and prints the average similarity scores between pairs of definitions.

2. Calculates and prints the average frequency of the most frequent word in pairs of definitions.

3. Creates two dictionaries to store word-average similarity pairs and word-average frequency pairs, respectively.

4. Adds the word and its average similarity score to the 'word_avg_similarity_dict'.

5. Adds the word and its average frequency score to the 'word_avg_freq_dict'.

6. Display the result.

7. Creates a PrettyTable to organize and display the aggregated results, showing the average similarity values for different pairs of definitions.

3 Conclusion

	Astratto	Concreto
Generico	pain 0.24	door 0.22
Specifico	blurriness 0.08	ladybug 0.60

Table 1: Word Overlap Approach

	Astratto	Concreto
Generico	pain 0.18	door 0.14
Specifico	blurriness 0.12	ladybug 0.15

Table 2: Word Frequencies Approach

The tables show that in this specific case, the Word Overlap Approach yielded higher similarity scores for "ladybug" and "pain" because these words share more overlapping words with other specific and abstract words, respectively. On the other hand, the Word Frequencies Approach produced higher scores for "door" and "ladybug" because these words have higher word frequencies within their respective categories.