TLN Part 2 - Concept Similarity

Paletto Andrea, Tuninetti André

September 5, 2023

Contents

1 Introduction

1.1 Task

The exercise involves implementing three similarity measures based on WordNet.

- Wu & Palmer similarity
- Shortest Path similarity
- Leacock & Chodorow similarity

For each similarity measure, we then normalize the results and calculate the Spearman correlation index and the Pearson correlation index against the 'target' values present in the annotated file. We then compare our results to the ones obtained using NLTK built in functions.

2 Methodology

In every scenario we computed the similarity measures with the function provided by the NLTK library as well as our own custom made functions. The same concept applies to the Spearman's and Pearson's correlation coefficients where we used our own implementations alongside the ones provided by the Scipy library

2.1 Utility functions

The following utility functions are implemented to assist the functions that compute the similarities.

The `get_depth(syn)` function calculates the depth of a given Synset to its topmost hypernym.

The `find_lcs(s1, s2)` function finds the Lowest Common Subsumer (LCS), which is the most specific common ancestor, between two WordNet Synsets, `s1` and `s2`. The process involves comparing the hypernym paths of `s1` and `s2` to identify a common synset that has the maximum depth.

The `get_len(s1, s2)` function computes the minimum path length between any pair of synsets. It iterates on the hypernym paths of the synsets until it finds a common ancestor, then computes the length by adding each distance.

The `get_max_depth()` function returns the maximum depth in WordNet. It uses a list comprehension to obtain the depths of all synsets and then finds the maximum depth.

2.2 Wu-Palmer Similarity Calculation Functions

The following functions are used to compute the Wu-Palmer concept similarity scores between pairs of WordNet Synsets:

- The `max_stock_wp(syn_arr_w1, syn_arr_w2)` function finds the maximum Wu-Palmer Similarity between two lists of synsets using NLTK.
- The `custom_wp(s1_arr, s2_arr)` function calculates the Wu & Palmer similarity score between pairs of synsets from two arrays.
- The `compute_wp(df)` function handles the process of computing the custom Wu-Palmer similarity between the terms retrieved from the CSV file.
- The `display_wup(wp_arr, target_arr, ref_wup_arr, df)` function displays the results in a formatted table.
- The `show_wup_correlations(target_arr, wp_arr)` function displays Pearson and Spearman correlation values between the custom Wu-Palmer similarity values and the target values. To get some context it also computes and displays the correlation values using the SciPy library.
- The `demo_wup()` function acts as a pipeline controller for the Wu & Palmer similarity part of the exercise.

2.3 Shortest Path Similarity Calculation Functions

The following functions calculate similarity scores between arrays of synsets:

- The `stock_sp(syn_arr_w1, syn_arr_w2)` function computes the similarity between two arrays of synsets using NLTK.
- The `shortest_path(syn_arr_w1, syn_arr_w2)` function computes the shortest path similarity between two arrays of synsets.
- The `compute_sp(df)` function handles the process of computing of Shortest Path similarity scores between synsets.
- The `display_sp_table(sp_arr, target_arr, df)` function displays a formatted table with calculated custom Shortest Path similarity values.
- The `show_sp_correlations(target_arr, sp_arr)` function displays Pearson and Spearman correlation values between the custom Shortest Path similarity scores and the target values. To get some context it also computes and displays the correlation values using the SciPy library.
- The `demo_sp()` functionacts as a pipeline controller for the Shortest Path similarity part of the exercise.

2.4 Leacock-Chodorow Similarity Calculation Functions

The following functions calculate the Leacock-Chodorow similarity between two synsets:

- The `max_stock_lc(syn_arr_w1, syn_arr_w2)` function finds the maximum Leakcock-Chodorow similarity between two lists of synsets using NLTK.
- The `leakcock_chodorow(s1_arr, s2_arr)` function calculates and returns the Leacock-Chodorow similarity score between two synsets.
- The `compute_lc(df)` function handles the process of computing of Leakcock-Chodorow similarity scores between synsets.
- The `display_lc_table(lc_arr, target_arr, df)` function displays a formatted table with calculated custom Leakcock-Chodorow similarity values.
- The `show_lc_correlations(target_arr, lc_arr)` function displays Pearson and Spearman correlation values between the custom Leacock-Chodorow similarity scores and the target values. To get some context it also computes and displays the correlation values using the SciPy library.
- The `demo_lc()` function acts as a pipeline controller for the Leacock-Chodorow similarity part of the exercise.

2.5 Correlation Coefficient Functions

The following functions calculate correlation coefficients between two arrays using different methods.

2.5.1 Pearson Correlation

The `pearson_correlation(x, y)` function calculates the Pearson correlation coefficient between two arrays of values, `x` and `y`.

2.5.2 Spearman Rank Correlation

The `spearman_rank_correlation(x, y)` function calculates the Spearman rank correlation coefficient between two arrays of values, `x` and `y`.

3 Conclusions

The correlation cohefficients computed using Scipy's built in functions differ from the result obtained using our custom implementation by a factor of 0.01. This is probably due to some heuristic implemented in Scipy. In most of the cases our implementation returns the same values as the NLTK similarity functions. In a few occasions the results differ by a substantial margin, this is explained

by the fact that we compute the LCS in different ways and sometimes we take different elements as LCS resulting in different depths and therefore different similarity scores.

3.1 Wu-Palmer Correlations

	Pearson Correlation	Spearman Correlation
Custom	0.25333143427210936	0.2937916278568271
Scipy	0.2526117426974728	0.29707114520466993

Table 1: Custom Wu-Palmer Correlation Scores

	Pearson Correlation	Spearman Correlation
Custom	0.31037102559519314	0.3449862888790277
Scipy	0.30948928972702505	0.3481134284103167

Table 2: Stock Wu-Palmer Correlation Scores

3.2 Shortest Path Correlations

	Pearson Correlation	Spearman Correlation
Custom	0.23524787274719886	0.2918703537916206
Scipy	0.23457955492689442	0.3024705948512329

Table 3: Custom Shortest Path Correlation Scores

	Pearson Correlation	Spearman Correlation
Custom	0.3710271367945775	0.292343384275022
Scipy	0.369973082428684	0.302884609000026

Table 4: Stock Shortest Path Correlation Scores

3.3 Leakcock-Codorow Correlations

	Pearson Correlation	Spearman Correlation
Custom	0.32814851949249974	0.2918703537916206
Scipy	0.32721627938030495	0.3024705948512329

Table 5: Custom Leakcock-Codorow Correlation Scores

	Pearson Correlation	Spearman Correlation
Custom	0.342314792538021	0.3033076669794824
Scipy	0.34134230733194704	0.31050844567427255

Table 6: stock Leakcock-Codorow Correlation Scores