TLN Part 3 Lab 07 - False Friends

Paletto Andrea, Tuninetti André

September 12, 2023

Contents

1 Introduction

In this document, we present the implementation of a solution to identify pairs of words that share similar or identical spellings but have different semantic meanings in different languages. This task is commonly referred to as identifying "false friends". We aim to detect potential false friends between English and Spanish words by leveraging techniques such as Levenshtein distance and word embeddings.

2 Methodology

2.1 Function Overview

2.1.1 words_overlap(word1, word2)

This function calculates the Levenshtein distance between two words and returns `True` if the distance is less than or equal to a threshold (half of the maximum word length). This function helps identify words with similar spellings.

2.1.2 count_overlapping_words(meaning1, meaning2)

This function counts the number of overlapping words between two meanings. It tokenizes and normalizes the meanings to lowercase before counting overlapping words.

2.1.3 is_false_friend(word1, word2)

This function determines whether a pair of words is a potential false friend. It performs the following steps:

- Checks if the words have overlapping spellings using `words_overlap`.
- Retrive the meaning using 2 different approaches:
  - Retrieving the meanings of the words in English and Spanish using PyMultiDictionary.
  - Retrieving the meanings of the words in English and Spanish using Babelnet.
- Normalizes the meanings to lowercase and computes the semantic similarity between them using word embeddings.
- Considers words potential false friends if they have similar spellings or if their semantic similarity is below a threshold (0.80).

2.2 Workflow

The workflow of the code involves these main steps:

1. Read pairs of English and Spanish words from a CSV file.
2. Create tuples to store the word pairs.
3. Iterate through the word pairs and determine if they are potential false friends using the `is_false_friend` function.
4. Print the results, indicating whether the word pairs are potential false friends or not.

3 Conclusion

In this section, we analyze the most relevant cases.

3.1 Case 1: Parent / Pariente

In this case, the PyMultiDictionary approach correctly categorizes the terms as "Potential False Friends," while the BabelNet approach erroneously categorizes the tuple as "Not False Friends." This discrepancy arises because the tokenized BabelNet definition similarities, with a value of 0.84, surpass our chosen threshold of 0.80. Conversely, the PyMultiDictionary approach yields a similarity score of 0.74, falling below this threshold.

3.2 Case 2: Accident / Accidente

In this case, both the PyMultiDictionary and the BabelNet approaches erroneously categorize the tuple as "Not False Friends." This discrepancy occurs because the tokenized BabelNet definition and pydictionary similarities exceed the chosen thresholds: 0.73 for BabelNet and 0.74 for PyDictionary.

3.3 Case 3: Simpathy / Simpatia

In this case, while using the PyMultiDictionary, we encountered certain limitations. Specifically, the meaning of 'Simpatia' was not accessible within the resource. In such situations, we make the assumption that the words are, in fact, False Friends. On the other hand, when utilizing Babelnet, we can compute a similarity score of 1, indicating a perfect match, therefore the words are not false friends.