TLN Part 3 Lab 02 - Content2Form

Paletto Andrea, Tuninetti André

September 12, 2023

Contents

1 Introduction

In this exercise we implemented a content-to-form experiment utilizing WordNet definitions. The goal of this experiment is to implement onomasiologic search using the provided definitions of the terms : "door", "ladybug", "pain", and "blurriness".

2 Methodology

2.1 Function overview

We defined a few utility functions for word preprocessing:

1. **remove_punctuation(sentence)**: This function removes punctuation characters from a sentence except for hyphens.
2. **clean_and_split(text, stopwords)**: This function cleans and processes text by removing punctuation, splitting into words, converting to lowercase, lemmatizing, and excluding stopwords.
3. **get_words_frequency(words)**: This function calculates the frequency of words.
4. **get_hyponyms(word)**: This function retrieves the hyponyms of a given word using WordNet from NLTK.
5. **signature(sense, stop_words)**: This function computes the signature of a WordNet synset, including its name and example words.
6. **computeOverlap(signature, context)**: This function calculates the overlap between two sets of words.
7. **modifiedLesk(words, disambiguation_context, stop_words)**: This function implements a modified version of the Lesk algorithm to find probable synsets for candidate genera.

2.2 Workflow

For each of the words **door, ladybug, pain,** and **blurriness,** we perform the following steps:

1. **Load data**: We start by importing the necessary libraries and loading the definitions from a .tsv file. The definitions are stored in a dictionary named 'definitions', where the keys are words, and the values are lists of their corresponding definitions.
2. **Clean and preprocess each definition**: We then clean and preprocess each word definition. This involves removing punctuation, splitting the text into individual words, converting them to lowercase, lemmatizing, and filtering out stopwords. These cleaned words are stored in the 'clean_words' list.
3. **Create a disambiguation context**: To disambiguate the word senses effectively, we combine all the cleaned word sets from step 1 to create a disambiguation context. This context represents the words associated with the specific word being analyzed.
4. **Calculate word frequencies**: We compute the frequency of each word in the cleaned definitions. This information helps to identify which words are most commonly associated with the word of interest.
5. **Select candidate genera**: The top 15 most frequent words from the cleaned definitions are selected as candidate genera. These words serve as potential indicators of the word's sense.
6. **Modified Lesk Algorithm**: To determine the most likely senses of a word, we employ a modified Lesk algorithm. This algorithm relies on the concept of "genus." It takes into account the disambiguation context, which includes the candidate genera, in order to identify the sense that shares a more specific relationship with those candidate genera. In WordNet, this relationship corresponds to "hyponyms," which are more specific terms within a given semantic field. By identifying the sense with the most common hyponyms with the candidate genera, the algorithm effectively disambiguates word senses based on the principle of genus. Using hyponyms allows for a more comprehensive representation of a word's meaning, capturing its specific sense in a particular context (via hyponyms).

3 Conclusion

We achieved expected results with this strategy when dealing with concrete concepts:

- By utilizing the definitions of the term **door** we obtained a high score of 9 for the synset **doorway.n.01.** Meanwhile, when exploring the term **ladybug** we encountered a tie between **four-lined_plant_bug.n.01** and **ladybug.n.01** synsets, both scoring 5.

However, things change when we process abstract concepts:

- Utilizing the definitions of the term **pain** we identified **bad.s.03** with a score of 7 and **experience.v.03** with a score of 6. Additionally, there was a tie between **affection.n.01** and **agony.n.01,** both receiving a score of 5.
- When we examined the word **blurriness** we encountered another tie, this time it was between **accommodating_lens_implant.n.01, collage.n.01,** and **picture.n.01,** all scoring 6, followed by **acuity.n.01** with a score of 5.

It's interesting that in both instances involving abstract concepts, the results include antonyms of the words we were searching for, such as **affection** when processing the definitions of **pain** and **acuity** when processing the definitions of **blurriness**.