Clean Coders (Sankalp Bhandari, Shraddha Bang)

UNIVERSITY OF TEXAS AT DALLAS | 800 W CaMPBELL RD RICHARDSON TX 75080

Semantic Textual Similarity

Natural Language Processing



Table of Contents

[Problem Description 2](#_Toc26204520)

[Proposed Solution 2](#_Toc26204521)

[Full Implementation Details 3](#_Toc26204522)

[Pre-Processing 3](#_Toc26204523)

[Similarity Measures (Feature Description) 3](#_Toc26204524)

[Structural Similarity Measures 3](#_Toc26204525)

[Syntactic Similarity Measures 4](#_Toc26204526)

[Semantic Similarity Measures 4](#_Toc26204527)

[Programming Tools 5](#_Toc26204528)

[Architectural Diagram 5](#_Toc26204529)

[Result and Error Analysis 5](#_Toc26204530)

[Problems faced 5](#_Toc26204531)

[Pending issues 5](#_Toc26204532)

[Potential Improvement 5](#_Toc26204533)

# Problem Description

For this project, we have designed and implemented a model that determines how similar two chunks of text are. The similarity score takes an integer value between 1 and 5 (included). The higher the score, the more similar the two chunks are.

In general, semantic textual similarity (STS) is a challenging problem; as it requires both an understanding of lexical-level similarity, and the semantic composition of the two chunks of text being analyzed. As a reference, here are some motivating examples:

*Sentence 1: Birdie is washing itself in the water basin.*

*Sentence 2: The bird is bathing in the sink.*

*Score: 4*

*Comment: Both sentences convey the message that a bird is taking a bath.*

*Sentence 1: The young lady enjoys listening to the guitar.*

*Sentence 2: The young lady enjoys playing the guitar.*

*Score: 2*

*Comment: Both sentences involve a lady and a guitar, but convey different actions i.e. listening to the guitar and playing the guitar respectively.*

# Proposed Solution

For our proposed solution, we aim to quantify the similarity of pairs of sentences by encoding a variety of relatedness features in a vector of attributes and then predicting their similarity scores by employing machine-learning algorithms. We have measured the similarity of sentences from three perspectives - structural, syntactical, and semantic. We have chosen to approach the estimation of similarity as a classification problem. Hence, we use the quantified similarity of sentence pairs to train a classifier that can then be applied to predict similarity score for the unseen pairs.

# Full Implementation Details

## Pre-Processing

1. The sentences are read line by line from the file.
2. Each line is split by tab to extract the id, two sentences to be compared and the label.
3. The two sentences are converted into lower case and punctuations are removed.
4. The dependency parse tree of those two sentences is calculated using spacy library.
5. The sentences are then tokenized using the wordnet.
6. The stop words are then removed from the list of tokens
7. POS tag of each of those tokens is found using wordnet
8. Each token is then lemmatized using the token and the POS tag of that token using wordnet library
9. The wordnet features like hypernyms, hyponyms, holonyms, meronyms and synsets are found are each of these lemmas

## Similarity Measures (Feature Description)

### Structural Similarity Measures

#### Ratio of Sentence Lengths **:**

The relative length of two sentences (length of smaller sentence over the longer one) provides a simple measure of similarity. However, this naïve attribute of a pair can be useful when combined with other more conceptual measures.

#### POS tag relative length

The relative length of POS tags (like nouns, verbs, adj, adv) are also calculated and serve as feature as they give the structural information about the two sentences.

#### Dependency parse tree comparison

We used Spacy to extract the dependency parse tree. From this, we extracted the subject(nsubj), predicate(root) and object(dobj) of the sentences and calculated the best score mechanism to find the similarity between the two nsubj, roots and dobj with best sense.

#### Word alignment

For this feature,we derive the word embedding using bag of word model approach. We then use “difflib’s SequenceMatcher” to find out the sequence matching score between two embeddings of a pair of sentences.

### Syntactic Similarity Measures

#### Bag of words + Jaccard Similarity

A simple measure for computing the similarity of a sentence pair is the number of words they have in common. Hence a sentence can be considered as a set of words. To incorporate this perspective, we calculate the Jaccard similarity coefficient of a pair of sentences

#### Bag of words + Cosine Similarity

The above same bag of word model can be used to formulate the vectors for each sentence and calculate the cosine similarity of these vectors. The cosine similarity is advantageous because even if the two similar sentences are far apart by the Euclidean distance (due to the size of the sentences), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

### Semantic Similarity Measures

#### Bag of words + Synsets + Jaccard Similarity

Other sets of vocabulary-based similarity measures can be devised by getting all the synonyms of each word of sentences and considering them in the comparison process. One of these measures can be calculated by applying WordNet for obtaining synonyms of words. For this WordNet synonymy measure, the corresponding synsets of all the lemmas of the effective words in sentences are retrieved from WordNet. The sets of synsets of a pair of sentences are then compared to each other and the jaccard similarity is calculated.

#### Synsets + Jaccard Similarity

Only the synsets extracted from the wordnet are taken into consideration and the jaccard similarity is calculated on that sets as they will have more features in common and the lemmas will not get considered here. This proved a good measure. As the set of synsets of the similar lemmas will be same and they will give more semantic similarity than the lemmas themselves.

#### Bag of words + Synsets + Cosine Similarity

As explained earlier, in order to have more semantic features, the vectors are now generated taking the synsets of effective words and their cosine similarity is calculated.

#### Synsets + Cosine Similarity

Here vectors are generated using the synsets only and their cosine similarity is calculated.

#### Word sense-based similarity

This measure uses a WordNet-based word sense disambiguation approach to find the best senses of tokens between the sentences. We have used path similarity measure from wordnet for this approach . We calculate the best score based on the highest path similarity for each token from first sentence among tokens from other sentence.

#### POS Tag based similarity

For this similarity measure, we get the POS tags of each word in the subject and object phrases of a sentence and form a set of these tags. Then we again use the best score mechanism explained in above feature for these sets of tags.

#### Word Mover Distance

WMD enables us to assess the “distance” between two sentences in a meaningful way, even when they have no words in common. It uses Google’s pretrained vector embeddings of words. The intuition behind the method is that we find the minimum “traveling distance” between documents, in other words the most efficient way to “move” the distribution of sentence 1 to the distribution of sentence 2. It been shown to outperform many of the state-of-the-art methods in *k*-nearest neighbors classification.

## Programming Tools

* 1. [spaCy](https://github.com/explosion/spaCy): Python API commonly used in the industry
  2. [scikit-learn](https://scikit-learn.org/stable/): Python API for ML frameworks
  3. [NLTK](https://www.nltk.org/): Python API for common NLP tasks
  4. Google’s pre-trained Word2Vec model
  5. Difflib’s SequenceMatcher

## Architectural Diagram

## Result and Error Analysis

## Problems faced

## Pending issues

## Potential Improvement