Semantic Textual Similarity Assignment

SemEval 2012: The First Joint Conference on Lexical and Computational Semantics

Students:

- Mario Rosas
- Alam Lopez

Lab Professor: Salvador Medina Herrera

Introduction

The current assignment is intended to use all the concepts acquired in the IHLT class, by replicating the 6th task of the 2012 SemEval.

Task to accomplish: Compute the Semantic Textual Similarity (STS), between various pairs of sentences. In a range from 0-5.

- **(5) Completely equivalent**, as they mean the same thing
- **(4) Mostly equivalent**, *but some unimportant details differ.*
- **(3) Roughly equivalent**, *but some important information differs*
- **(2) Not equivalent**, *but share some details.*
- **(1) Not equivalent**, *but are on the same topic.*

We will implement different approaches:

- a) Explore some lexical dimensions.
- b) Explore the syntactic dimension alone.
- c) Explore the combination of both previous.
- d) Add new components at your choice

Finally, compute the pearson correlation between the predicted similarity versus the ground truth/gold standard.

p-correlation baseline:0.31

General Steps

We decided to merge the first the approaches a, b, c into the same workflow, and with the insights obtained take the most of them to tackle d)

We develop all the needed code in separated methods to be called

1. Create a method to load the data in a more efficient way.
2. Develop the techniques to be used and the similarity metrics.
3. Write a main code to test approaches by calling the methods.
4. Analyze the results and get insights
5. Create a ML regressor using the insights

1. Create a method to load the data in a more efficient way.

We wrote a python method to read all the datasets and load them all into the same variable, considering both, train and test, and the pair of words with its corresponding ground truth value.

```
# Slide 1
class Dataset():
    def __init__(self, path):
        self.path = path
        self.files = [file for file in os.listdir(path) if 'STS' in file]
        self.inputs = {file.split('.')[-2]:file for file in self.files if file.split('.')[1] == 'input'}
        self.labels = {file.split('.')[-2]:file for file in self.files if file.split('.')[1] == 'gs'}

    def __getitem__(self,dataName = 'all'):
        dataname  = ['SMTeuroparl','MSRvid', 'MSRpar'] if dataName == 'all' else dataName
        dataname = dataname if isinstance(dataname, list) else [dataname]
        return pd.concat([self.read_as_df(self.inputs[name], self.labels[name]) for name in dataname], ignore_index=True)
```

2. Develop the techniques to be used and the similarity metrics.

We developed the code for each of the techniques, such as lemmatization, PoS, WSD, etc. Also to compute the similarity, and be able to eacily change from one to another.

```
def jaccard_distance(self, sentence1, sentence2):
        return 1 - (len(sentence1.intersection(sentence2))/len(sentence1.union(sentence2)))

    def dice_distance(self, sentence1, sentence2):
        return 1 - ((2*len(sentence1.intersection(sentence2)))/(len(sentence1)+len(sentence2)))

    def overlap_distance(self, sentence1, sentence2):
        return 1 - (len(sentence1.intersection(sentence2))/min(len(sentence1),len(sentence2)))
```

3. Write a main code to test approaches by calling the methods.

We decided to compute the techniques and similarity metrics for all the train datasets separated.

```
for dataname in ['SMTeuroparl', 'MSRvid', 'MSRpar', 'all']:
    dt = train[dataname]

    # ----- Tokenization -----
    # NLTK
    dt[2] = tp.tokenize_data(list(dt[0]),'nltk')
    dt[3] = tp.tokenize_data(list(dt[1]),'nltk')
    # spaCy
    dt[4] = tp.tokenize_data(list(dt[0]),'spacy')
    dt[5] = tp.tokenize_data(list(dt[1]),'spacy')
```

4. Analyze the results and get insights

The overall max p-correlation was 0.65

We developed in step 1, a method to show the results and get a clearer understanding on how the pearson correlations were across the techniques, datasets and similarity metrics. We decided that the besy approach was a heatmap plot with the fixed color range from 0 to 1.

Therefore, there are four heatmaps in the report. One for each separated dataset, and one for all of them together.

5. Create a ML regressor using the insights

The overall max p-correlation was 0.6719

We decided to use the MLP Regressor from Scikit learn library as our regressor, in order to train our system with the features defined and predict the similarity of test sentences.

Conclusions

The results of the Pearson correlation coefficient analysis (statistic=0.6719, p-value=1.979e-257) indicate a strong linear relationship between the predicted values and the actual values, demonstrating the effectiveness of the Multi-Layer Perceptron (MLP) regressor fitted by grid-search cross-validation. In particular, it is interesting to contrast these results with the alternative approach of using Dice or Cosine similarity measures directly. While the MLP model excels at capturing nuanced relationships, direct employment of Dice or Cosine similarity tends to yield lower but closely aligned correlation results.