Semantic Text Similarity

By:
Andrey **Núñez**
Eric **Walzthöny**

Problem

Determine the similarity between two sentences. One sentence is said to be "paraphrased" when the content (or message) is the same, but uses different words and or structure.

The bird is bathing in the sink.

Birdie is washing itself in the water basin.

Pipeline

1. Data Loading
2. Pre-processing
3. Feature Extraction: Standard
4. Feature Extraction: Additional
5. Post-Processing
6. Modeling
7. Evaluation

Data Loading

Two functions were built to load the separate files:

- STS.input.*
- STS.gs.*

Pre-processing

- Simple processing sequence:
  - Removal of:
    - Punctuations (string.punctuation)
    - Stopwords (nltk.corpus.stopwords('english').words)
  - Tokenization:
    - Sentence tokenization
    - Word tokenization
  - Tagging:
    - POS-tag
  - Lemmatization
    - Reduce inflected word form
    - Usage of *penn2morphy* (converts from Stanford to Wordnet tag)

Feature Extraction: Standard

Using metrics used in the assignments:

- Jaccard Distance
- Synset similarity
  - Path lengths between concepts:
    - Wu-Palmer, Leacock & Chodorow, path
  - Information concept:
    - Lin (added brown_ic, LCS)
- N-gram similarity
  - Uni-, Bi-, Tri-
- Named Entities
- Length of sentences

Feature Extraction: Additional

Other features which were used:

- Sentence Length
- FuzzyWuzzy string matching
- Levenshtein edit distance / ratio

Post-processing

- Standardization of the data by z-score
    - $z = x - \mu / \sigma$

- This was the only post-processing step that was done.

Modeling

- Treating it like a regression problem
- Using SVR in as shown in literature it works best
- Fine-tuning of model and hyperparameter optimization with GridSearch

Evaluation

- Adding / Removing features to see effect on performance
- We were able to achieve: **0.7892 (pearsonr)**

Conclusion

- Insights:
    - Choosing features is important and depends on the type of problem one is working on.
    - NLP varies, and needs a lot of information.
        - Information can be lost
        - Information might be from latent variables → embedding
    - Extracting the right features is a tedious task
        - Trial and error
- What we didn't expect?
    - Our trials showed that Levenshtein and FuzzyWuzzy string matching did not benefit the performance of the SVM.
    - Most important features were similarity measures (WuP, Lin, Path, LCH)
    - N-grams provided a substantial improvement to the model (~5 - 10%)

Further Works / Thoughts

- We'd like to build on this idea and perform:
  - Ensemble method with different models
  - More precise GridSearch, with more options for the parameters
  - Feature Engineering:
    - Are there combinations of features which can provide more information to the model than standalone?
  - Different Approaches:
    - CNN, RNN, Siamese Networks and Transformers (Embeddings)
  - Data Augmentation:
    - Hard without having a GS available to check
    - Use Transformers-based to corroborate the GS to generate more training data
      - AEDA, Multi-Round Text Augmentation
  - Effect of different post-processing standardizations / normalization

Appendix 1: Levenshtein edit distance / ratio

The Levenshtein distance between two strings a, b (of length $|a|$ and $|b|$ respectively) is given by $\operatorname{lev}(a, b)$ where

$$\operatorname{lev}(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} \operatorname{lev}(\operatorname{tail}(a), b) \\ \operatorname{lev}(a, \operatorname{tail}(b)) \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) \end{cases} & \text{otherwise.} \end{cases}$$

Appendix 2: FuzzyWuzzy ratio

The Levenshtein distance between two strings a, b (of length $|a|$ and $|b|$ respectively) is given by $\operatorname{lev}(a, b)$ where

$$\operatorname{lev}(a, b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) & \text{if } a[0] = b[0] \\ 1 + \min \begin{cases} \operatorname{lev}(\operatorname{tail}(a), b) \\ \operatorname{lev}(a, \operatorname{tail}(b)) \\ \operatorname{lev}(\operatorname{tail}(a), \operatorname{tail}(b)) \end{cases} & \text{otherwise.} \end{cases}$$