Semantic Textual Similarity

Semantic Textual Similarity (STS) quantifies the extent to which two text snippets share the same meaning. Given a pair of sentences, the model assigns a similarity score on a continuous scale from 0 to 1, where 0 denotes no semantic overlap and 1 indicates full equivalence. The model's performance is evaluated by calculating the correlation between its similarity scores and human judgments.

Team - Player No 456

Sasidhar Chavali - 2021101111
Rohit Gowlapalli - 2021101113
Abhinav Reddy Boddu - 2021101034

Datasets

STS Benchmark
Quora Dataset
Paraphrase Corpus (by microsoft)
SICK Dataset

Methodology

1. **N-gram Based Statistical Methods**
   We begin with simple statistical techniques that compare text based on overlapping words or sequences of words (n-grams). These methods provide a straightforward way to measure similarity.
2. **Neural Network Models (CNN, RNN, LSTM)**
   To better understand sentence structure and meaning, we use neural networks. CNNs help identify important word patterns, while RNNs and LSTMs capture the order and relationships between words, making them useful for text similarity.
3. **Embedding-Based Approaches**
   Instead of comparing words directly, we use word & sentence embeddings dense vector representations that capture meaning. Models like Word2Vec, FastText & GloVe help represent words in a meaningful way, while SBERT improves sentence understanding.
4. **Transformer-Based Models**
   We leverage advanced transformer models like RoBERTa, DeBERTa, and T5, which are designed to understand context and meaning deeply. These models analyze entire sentences at once, making them more effective for capturing semantic relationships.
5. **TF-IDF with Cosine Similarity**
   We also use a classic text comparison method where words are weighted based on their importance in a document (TF-IDF), & similarity is measured using cosine similarity.
6. **Ensemble Methods**
   To improve accuracy, we combine multiple models using ensemble techniques. Boosting methods like Adaboost and Gradient Boosting help refine predictions, while bagging techniques like Random Forest reduce errors by averaging multiple results.

Evaluation Strategies

To assess the effectiveness of our models in measuring Semantic Textual Similarity (STS), we use a mix of quantitative metrics and qualitative analysis.

1. **Pearson and Spearman Correlation**
   Since STS involves predicting similarity scores, we measure how well our model's predictions align with human-labeled scores using Pearson and Spearman correlation coefficients. These metrics evaluate the strength and direction of the relationship between predicted and actual values.
2. **Mean Squared Error (MSE) and Root Mean Squared Error (RMSE)**
   To analyze how far our predictions deviate from actual similarity scores, we calculate MSE and RMSE. Lower values indicate that our model makes more accurate predictions.

Timeline

Module 1: Setup & Data Exploration
In the first phase, we collect datasets, and clean the data. We then explore the data to understand sentence distributions and any imbalances.

Module 2: Classical & Deep Learning Models
Next, we train simple models like SVM and Random Forest using extracted features (like TF-IDF and n-grams). We also develop deep learning models with word embeddings (Word2Vec, FastText) and BiLSTM to see how they perform.

Module 3: Transformer Models & Ensemble Methods
We then fine-tune transformer models like BERT and T5 on our data. After that, we combine different models using techniques like stacking and averaging to boost performance and test their robustness. We also test basic methods like Jaccard Similarity and TF-IDF.

Module 4: Evaluation, Optimization & Reporting
In the final stage, we analyze the models' performance, make them more efficient, and test them on new data. We also prepare a report with all our findings.

Literature Review

- **ECNU**
  Combines three feature-engineered models (Random Forest, Gradient Boosting, XGBoost) and four neural networks using sentence embeddings' element-wise operations. Features include edit distance and tree kernels.
- **BIT**
  Measures semantic textual similarity using Jaccard-based information content. Incorporates WordNet hierarchy and combines sentence alignment with word embeddings for better results.
- **MITRE**
  An ensemble of five models, including SVM-based n-gram overlap, CRNNs, string similarity, word alignment, and an Enhanced BiLSTM model with attention and inference layers.
- **FCICU**
  Uses BabelNet for sense-based alignment, enabling cross-lingual semantic similarity computation. Employs string kernels and soft-cardinality for final similarity scores.
- **Compi_LIG**
  Integrates syntax-based, dictionary-based, context-based, and machine translation (MT)-based methods. Uses cosine similarity, weighted Jaccard, and monolingual alignment.
- **LIM_LIG**
  Enhances CBOW-based word embeddings with IDF and POS weighting. Sentence embeddings are derived from summed word vectors.
- **DT_Team**
  Uses alignments, sentence embeddings, and Gaussian Mixture Models. Preprocessing includes POS-tagging, NER, and lemmatization before computing similarity from aligned word-pairs.
- **Sent2vec**
  Unsupervised learning of sentence embeddings using compositional n-gram features. Computes similarity scores purely from cosine similarity of sentence embeddings.