Precog Task

Task chosen: 1 - Natural Language Processing

Participant Details:
Name - Sadhu Jay Vardhan
Education - BTech, Final year
Branch - Artificial Intelligence

My Sincere thank for giving us a chance to be a part of IIIT Hyderabad. I understand the importance of the task and the expectations associated with it, and I genuinely regret not being able to fulfill them completely. I assure you that this is not a reflection of my commitment or enthusiasm for the task but rather a result of unforeseen circumstances which included project submission and examination.

Programming Task

Problem Statement:

- To predict the similarity scores between pairs of words using SimLex-999 as test dataset.
- Obtain the similarity scores by converting words to numerical representations using ML algorithms like regression or classification. Model should be either unsupervised or semi-supervised where the entire SimLex-999 will be used as the test set.

Understanding the SimLex-999 dataset:

- word1, word2 are a pair of words.
- POS indicate Parts of Speech. "A" means adjective and likewise.
- SimLex999 is the human assigned similarity score ranges from 0 (no similarity) to 10 (perfect similarity).
- Assoc(USF) – Association measure based on the University of South Florida Free Association Norms is the Association measures quantify the strength of the relationship or connection between two words. A higher value suggests a stronger association. These measures can contribute to understanding the semantic relatedness of word pairs.
- SimAssoc333 - Similarity association score based on a different measure. It is similar to Assoc(USF) , this feature provides an alternative measure of the similarity or relatedness between words. Including multiple association measures can enhance the model's ability to capture nuanced semantic relationships.
- SD(SimLex) – Standard Deviation of SimLex scores. Standard deviation measures the amount of variation or dispersion in a set of values.

Procedure I have used to solve the problem using Cosine Similarity and K Mean Cluster:

This code performs several tasks related to natural language processing and word embeddings. Let me break down the main components and their functionalities:

1. Data Loading and Preprocessing:
   - SimLex999 data is loaded from a file named "SimLex-999.txt."
   - Brown Corpus data is loaded from a file named "brown.csv."

- The Brown Corpus data is preprocessed by tokenizing and converting text to lowercase. Stop words are removed, and the preprocessed text is stored in a new column named "preprocessed_text."

2. Word Embeddings using GloVe:
   - GloVe word embeddings (100-dimensional vectors) are loaded from the "glove.6B.100d.txt" file.
   - Word embeddings are generated for the preprocessed text in the Brown Corpus using the loaded GloVe embeddings. The embeddings for each word in a sentence are concatenated to form a matrix, and these matrices are stored in a new column named "word_embeddings."

3. Handling NaN Values:
   - Duplicates in the "preprocessed_text" column are removed.
   - Word embeddings are generated for the unique preprocessed text, and NaN values are handled by dropping rows with NaN values.

4. Cosine Similarity Calculation:
   - Cosine similarity matrices are calculated in chunks for subsets of the data using the word embeddings.
   - The cosine similarity matrix is converted to a DataFrame for better visualization.
5. Clustering using KMeans:
   - KMeans clustering is applied to the cosine similarity matrix with a specified number of clusters (20 in this case).
   - The clusters are assigned to the original data, and representative words for each cluster are displayed.

6. Average Similarity within Clusters:
   - Average similarity within each cluster is calculated and displayed.

7. Similarity Score Calculation:
   - A function is defined to calculate the average similarity between two words based on their clusters.

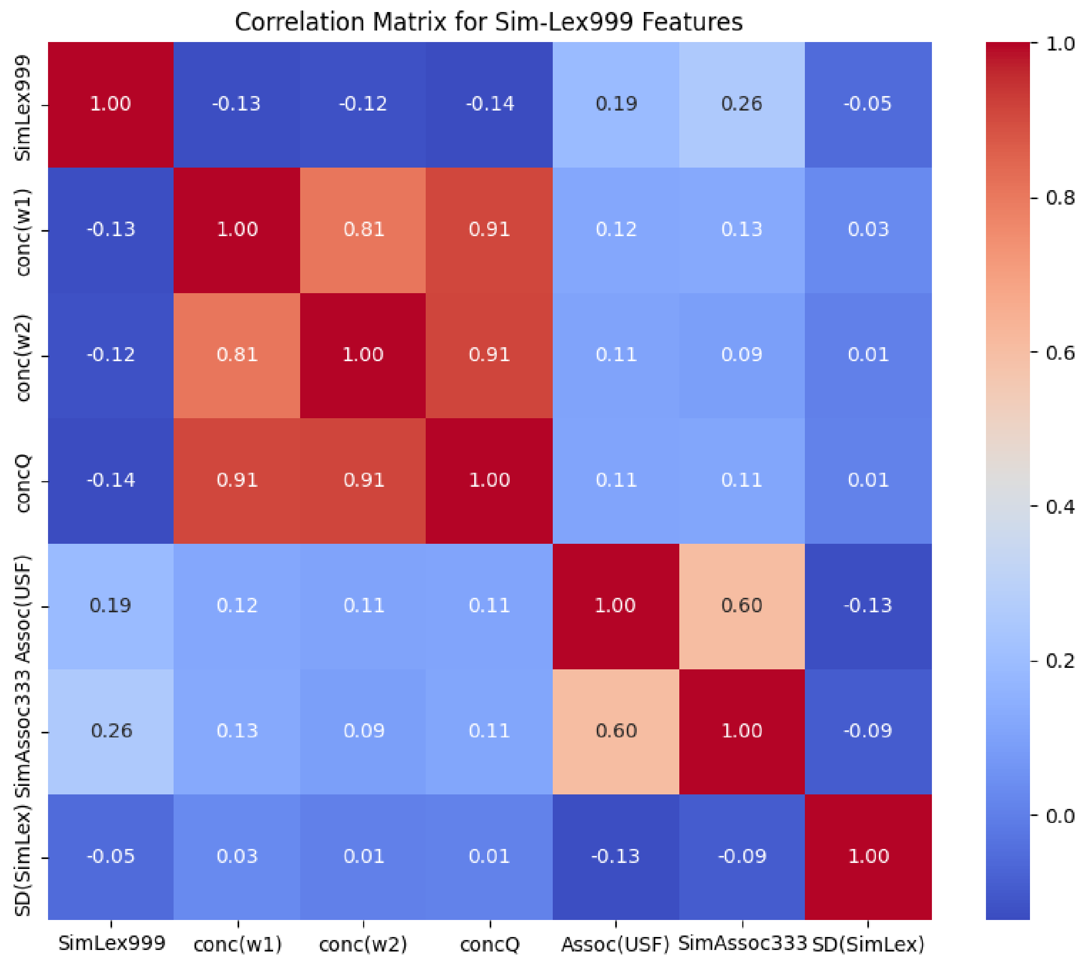
8. Evaluation with Spearman Correlation:
   - SimLex999 dataset is used for evaluation.
   - Predicted similarity scores are calculated using the defined function.
   - Spearman correlation is calculated between the actual SimLex999 similarity scores and the predicted similarity scores.

Reason for choosing GloVe over other pre-trained word embeddings?
I have chosen GloVe as it based on the global word-to-word co-occurrence. Keeping the large data set and computational constraints in mind I thought it would be a better approach.

Reason for using Clustering?

In the given question, we were given a choice of using either regression and clustering, initial approach was infact regression, I thought of training the model with cosine similarity values using linear regression first but unfortunately I could not as it was getting crashed (I used Colab Telsa T4 GPU). I even tried to process the code in chunks but was taking very long time. So I felt K means clustering for K=20(took 3 hours to implement).

In order to get the nearby values of SimLex999, we also need association values. I could not solve how to get those association values.



	SimLex999	conc(w1)	conc(w2)	concQ	Assoc(USF)	SimAssoc333	SD(SimLex)
SimLex999	1.00	-0.13	-0.12	-0.14	0.19	0.26	-0.05
conc(w1)	-0.13	1.00	0.81	0.91	0.12	0.13	0.03
conc(w2)	-0.12	0.81	1.00	0.91	0.11	0.09	0.01
concQ	-0.14	0.91	0.91	1.00	0.11	0.11	0.01
Assoc(USF)	0.19	0.12	0.11	0.11	1.00	0.60	-0.13
SimAssoc333	0.26	0.13	0.09	0.11	0.60	1.00	-0.09
SD(SimLex)	-0.05	0.03	0.01	0.01	-0.13	-0.09	1.00

Research Paper Task

The paper has initially stated with importance of BERTSCORE metric for evaluating natural language generation, particularly in machine translation and image captioning tasks. BERTSCORE is designed to address limitations in existing metrics like BLEU, which rely on surface-form similarity only. Instead, BERTSCORE utilizes pre-trained BERT contextual embeddings to compute the similarity between two sentences based on the cosine similarities between their tokens' embeddings. It spoke about the limitations of existing metrics n- gram-based metrics has where BERTSCORE aims to overcome. First, n-gram models often struggle with robustly matching paraphrases, leading to performance underestimation when semantically-correct phrases are penalized for differing from the surface form of the reference. BERTSCORE, by using contextualized token embeddings, is shown to be effective in addressing this issue. Second, n-gram models fail to capture distant dependencies and penalize semantically-critical ordering changes, which contextualized embeddings are better at handling. The paper spoke about the experiments conducted with BERTSCORE on machine translation and image captioning tasks involve correlating BERTSCORE and related metrics with human judgments. The results demonstrate that BERTSCORE correlates highly with human evaluations, showing stronger correlations in machine translation tasks compared to existing metrics like BLEU. In image captioning, BERTSCORE outperforms SPICE, a popular task-specific metric. Additionally, the paper discusses the robustness of BERTSCORE on an adversarial paraphrase dataset (PAWS) and shows that it is more robust to adversarial examples than other metrics

BERTScore in detail:

- For Token Representation, BERTScore uses contextual embeddings to represent the tokens in the input sentences. Contextual embeddings, such as BERT and ELMO, generate different vector representations for the same word in different sentences depending on the surrounding words, which form the context of the target word. The models used to generate these embeddings are most commonly trained using various language modeling objectives, such as masked word prediction.

- For Similarity Measure, BERTScore computes the similarity between two sentences as a sum of cosine similarities between their tokens' embeddings. The cosine similarity of a reference token and a candidate token is computed using the inner product of their pre-normalized vectors. While this measure considers tokens in isolation, the contextual embeddings contain information from the rest of the sentence.

- BERTScore optionally weights the cosine similarities with inverse document frequency (IDF) scores to downweight common words that are less informative. IDF scores are computed based on the frequency of each token in a large corpus of documents.

- BERTScore uses a baseline rescaling method to normalize the scores across different sentence pairs. This method involves subtracting the average score of the candidate sentence from the score of each token in the candidate sentence and adding the average score of the reference sentence. This rescaling method ensures that the scores are centered around zero and have a similar range across different sentence pairs.

The paper also includes the experimental setup:

- Contextual Embedding Models:
  1. Twelve pre-trained contextual embedding models are evaluated, including variants of BERT, RoBERTa, XLNet, and XLM.
  2. Models such as RoBERTa-large for English tasks, BERTchinese for Chinese tasks, and cased multilingual BERTmulti for other languages are used.
  3. Contextual embedding models generate representations at every layer in the encoder network, with intermediate layers considered more effective for semantic tasks.
- Machine Translation:
  1. Evaluation is performed on the WMT18 metric evaluation dataset, which includes predictions from 149 translation systems across 14 language pairs.
  2. Segment-level and system-level human judgments are available.

3. BERTSCORE is evaluated against other metrics using absolute Pearson correlation (|ρ|) and Kendall rank correlation (τ).
4. Significance testing is conducted using the Williams test for |ρ| and bootstrap re-sampling for τ.
5. Model selection experiments involve randomly sampling hybrid systems and comparing metric rankings with human rankings.

- Image Captioning:
  1. Human judgments from the COCO 2015 Captioning Challenge are used, with twelve participating systems.
  2. Metrics such as BLEU, METEOR, ROUGE-L, CIDER, BEER, EED, CHRF++, CHARACTER, SPICE, and LEIC are compared.
  3. BERTSCORE is computed with multiple references and compared with task-agnostic and task-specific metrics.
  4. Evaluation metrics include Pearson correlation with human judgments on two system-level metrics (M1 and M2).

On continuation, the paper spoke about the high performance of BERTScore. It consistently well as compared to RUSE, ITER, YiSi-1, R_{BERT}, F_{BERT} and other for tasks like Machine Translation, Image Captioning and many more. Also, BERTSCORE is relatively fast despite the usage of large pre-trained models.

The major strengths of the Research Paper:

- The paper introduces BERTScore, a novel automatic evaluation metric for text generation that addresses the limitations of existing metrics by leveraging contextual embeddings to measure token-level similarity between candidate and reference sentences. The research paper on BERTScore provides genuine information through benchmark results.
- The paper has also included comparisons with other evaluation metrics on experimenting on various tasks.
- The paper provides clear explanations and covered various functionalities of the targeted application. In this paper it spoke about token representation, similarity measure, importance weighting, and baseline rescaling used in BERTScore, enhancing the understanding of the metric's underlying principles.
-

The major Weakness of the Research Paper:

- Although I think the paper consist of good information, I don’t think It has got any weakness except for highlighting the strength of BERTScore and spoke very less about its weakness like complexity and other.

Three improvements:

- Making the discussion paragraph more crisp.
- Would use more visualization rather than tabular forms.
- Would give a glimpse on the applications mentioned.