RECRUITMENT TASK REPORT

PART1 (word similarity scores)

LIBRARIES USED:

- ***NumPy (Np)***
- ***spaCy***
- ***Pandas (pd)***
- ***Gensim***
- ***SciPy stats***

Steps followed:

1. Read the content of the file located at '/content/SimLex-999.txt'.
2. Display the content of the file.
3. Attempt to read the file content as a CSV file using pd.read_csv,
4. Manually create a Pandas DataFrame (df) with the provided data.
5. Download the "word2vec-google-news-300" model using gensim.downloader.
6. Use the downloaded Word2Vec model to calculate the similarity between two words ('absorb' and 'withdraw').
7. Load the 'en_core_web_lg' spaCy model.
8. Tokenize and preprocess the sentences from the DataFrame.
9. Train a Word2Vec model on the tokenized sentences.
10. Calculate the similarity score for each pair of words in the DataFrame.
11. Scale the similarity scores and add them to the DataFrame.
12. Use scipy.stats.spearmanr to calculate the Spearman correlation coefficient between the 'SimLex999' column and the scaled similarity scores.
13. Print the Spearman correlation coefficient.

PART2(phrase and sentence similarity)

Phrase similarity task :

Libraries used:

- ***datasets***
- ***pandas***
- ***spacy***
- ***numpy***
- ***scikit-learn***
- ***en-core-web-lg (spaCy model)***

Steps followed for execution:

1. Loaded the "PiC/phrase_similarity" dataset using the `datasets` library.
2. Preprocessed the text data, including tokenization, lemmatization, and lowercase conversion.
3. Split the dataset into training and testing sets using `train_test_split` from scikit-learn.
4. Utilized spaCy to obtain word embeddings for each phrase.
5. Calculated the cosine similarity between the embeddings to determine the similarity score.
6. Defined a threshold and classified the pairs as similar or not based on the similarity score.
7. Evaluated the model using accuracy, precision, recall, and F1-score.

Conclusion:

The model achieved an accuracy of approximately 49.69%, precision of 49.78%, recall of 70.25%, and an F1-score of 58.27%. These metrics indicate the performance of the model on the given text similarity task. Depending on the specific requirements of your application, you may need to adjust the threshold or explore different models to improve performance.

It's important to note that the success of the model depends on the quality and representativeness of the training data, as well as the chosen similarity metric and threshold. Further experimentation and tuning may be needed to optimize the model for your specific use case.

Sentence similarity task:

Libraries Used:

- ***datasets***
- ***pandas***
- ***pyarrow***
- ***spacy***
- ***scipy.spatial***
- ***scikit-learn***

Steps Followed for execution:

1. Data Loading: Gathered datasets using datasets and pandas, plus a Parquet file with pyarrow.
2. Text Preprocessing: Prepared text for analysis using spaCy's language model.
3. Sentence Embedding: Represented sentences as numerical vectors with spaCy's word vectors.
4. Similarity Calculation: Measured similarity between sentence pairs using cosine similarity.
5. Model Evaluation: Assessed performance with accuracy, precision, recall, and F1-score.
6. Conclusion: The model effectively identifies similar sentences, but precision could be improved. Potential for enhancement through optimization or advanced techniques.

*PRECOG **BERTscore** Paper Summary*

- Proposes BERTscore as a new metric for automatic evaluation of text generation
- Computes embeddings of reference and candidate texts using BERT
- Measures F1 score based on word-level pairwise similarity from embeddings
- Evaluates machine translation systems on WMT datasets
- Correlates highly with human judgments at both system and segment levels
- Outperforms existing metrics such as BLEU and ROUGE on these tasks
- Also evaluates image captions on COCO dataset, outperforming SPICE and BLEU
- Incorporates IDF weighting to account for token importance
- Uses greedy matching to compute recall, alternatives like WMD explored
- Experiments with BERT, XLNet, XLM, and RoBERTa contextual embeddings
- Selects optimal model layers based on development data correlation
- Provides code and pre-trained models for easy adoption and extension
- Allows integration into training objectives via differentiability of embeddings
- Evaluates English, Chinese and multiple European language translations
- Experiments on adversarial examples show robustness to semantic changes
- Proposes additional evaluation metrics for model selection tasks
- Super samples 10K hybrid systems from WMT datasets for analysis
- Ranks systems with metrics and computes accuracy vs. human judgments
- Measures mean reciprocal rank and difference to best system

- Replicates experiments on WMT16-17 datasets with other metrics
- Applies to abstractive text summarization data to capture meaning vs. grammar
- Finds recall correlates best with meaning, precision with grammaticality
- Ablation studies explore design choices like greedy vs. optimal matching
- Analyzes cases where BERTscore and BLEU disagree with human judgments
- BERTScore captures synonyms and word order flexibly unlike BLEU
- Discusses related work in embeddings, edit distances, learned metrics
- Contrasts approach with dependency on tools like parsers in MEANT
- Provides qualitative analysis of model behavior on examples
- Discusses future directions like training with BERTscore as differentiable loss
- Concludes BERTscore effectively captures human intuition for generation
- Slide 3:
- Open sources code and publishes pre-trained models
- Enables easy use for both research and production applications
- Provides avenue for tailoring evaluation to new domains and languages
- Inspires thinking of evaluation as iterative process paired with training
- Highlights general effectiveness of contextual embeddings for NLP
- Encourages exploration ofTask-specific metrics built on top of contextual embeddings
- Suggests directions like training models directly on expected evaluation
- Benchmarks help adoption by identifying best performing embedding variants
- Future work to design metrics capturing other desiderata like diversity

- Integrating BERTscore as part of training objective an interesting direction
- Experiments on more generation modalities like dialog, summarization
- Cross-lingual transfer learning could improve performance on low-resource languages
- Potential to apply insights from adversarial training to make metrics more robust
- Expand understanding of what aspects of generation humans truly value
- Continued progress on automatic evaluation will strengthen progress in text generation

Major Strengths:

1. The paper introduces BERTScore as a metric for evaluating text data, addressing drawbacks of existing evaluation metrics like BLEU and ROUGE.
2. The metric is robust and is effective against adversarial sentences, which shows its reliability in evaluating paraphrase adversaries.
3. The paper provides various evaluation methods for model selection tasks, offering a approach to system ranking and selection.

Potential Weaknesses:

1. While the paper shows the advantages of BERTScore over other metrics, it may benefit from a more comparative analysis with a wider range of evaluation metrics.
2. The paper heavily emphasizes the correlation of BERTScore with human judgments, potentially overlooking other important aspects of evaluation metrics such as diversity and fluency.
3. BERTScore's reliance on BERT and other contextual embeddings may introduce complexity and computational overhead, limiting its practicality in certain scenarios.
4. The paper's discussion of future directions and potential applications could be further expanded to provide a more comprehensive roadmap for leveraging BERTScore in various NLP domains.

Suggest three improvements to the paper, that would improve the paper?

Robustness:

- Test models and metrics with diverse texts: formal, informal, specialized.
- See how noise/errors affect evaluation.

Future Work:

- Suggest improvements for models and metrics.
- Recommend best practices for practitioners and researchers.

Missing Analysis:

- Deeper dive into why specific models/metrics excel or struggle.
- Consider characteristics of languages/pairs impacting performance