**Report: Text Representation and Semantic Similarity - Programming Task**

**INTRODUCTION**

This report gives a detail exploration of various programming techniques used to represent words, phrases, and sentences, along with measuring their semantic similarity. The tasks involved unsupervised learning approaches, leveraging pre-trained models, and investigating Large Language Models.

a. **Word Similarity Scores:** Given a pair of words, predict their similarity score. The focus is how do you convert a word to its numerical representation, on which learning algorithms (like Regression, classification etc) can be applied. Download the dataset from this link. You have to come up with an unsupervised / semi supervised method to achieve the task. Assume that you don't have any supervised training data at your disposal. The whole dataset will be used as a test set. Choose an appropriate metric that is suitable to assess the task and report the results. You have to come up with a solution for each of the following conditions:

1. Constraints on Data Resources: You can only use the following resources (any one or all) to solve the problem **(DON’T USE PRE-TRAINED MODELS!)** :

- any monolingual English corpus - Maximum 1 million tokens.

1. Unconstrained : Consider that the constraints above are removed and you are allowed to use any data or model.

Compare results/analysis across the two settings. What works, what doesn’t? And Why?

**Solution:**

In this task, we aim to predict the similarity score between pairs of words by converting each word into its numerical representation. The dataset provided will serve as our test set, and we will develop solutions under two different conditions: Constrained data resources and uncontrained resources. For the constrained part, we can use only a monolingual English corpus with a maximum of 1 million tokens and cannot use pre-trained models.

**Trial No. 1:**

In this trial, we enchance the one-hot encoding methods by incorporating Part-of-speech tagging. This helps in maintaining the semantic context of words, allowing us to differentiate between words with different meanings based on their POS tags. The words are then embedded considerin the POS tags and the similarity scores are computed.

**Trial No. 2:**

1. Data Preparation

Loading the SimLex-999.txt dataset and training 1 Million token Monolingual Corpus. Then extracted the word pairs and similarity scores from the SimLex-999 dataset. Next we normalize the similarity Score to range 0 and 1. Then e tokenize the corpus and pad sequence to ensure equal length.

1. Model Building

Neural Network Architecture

* + Input layers for the word sequence.
  + Embedding layers to convert words into dense vectors.
  + Bidirectional LSTM layers to process the sequence.
  + Concatenate LSTM outputs and pass them through dense layers with ReLU activation. And applied dropout regularization to prevent overfitting.
  + Output layer with a sigmoid activation function to ensure similarity scores in the range [0,1].

1. Model Compilation and Training

Compiling the model using the Adam optimizer and mean squared error loss function. Used early stopping to prevent outfitting. Validating the model’s performance on test data.

1. Model Evaluation

Evaluated the trained model performance on the test set using the test loss metric.

1. Inference Function

This function is used to predict similarity scores between arbitrary word pairs.

For Unconstrained Part we have use Word2Vec model, and have taken same datasets for comparison.

In Word2Vec, vectors are generated only for words present in the vocabulary or corpus. To obtain a vector for a new word, it needs to be added to the vocabulary, and then its embedding can be derived by training the Word2Vec model on the corpus, ensuring all words have corresponding embeddings. Essentially, Word2Vec generates embeddings for words based on their occurrence in the corpus, and adding new words involves updating the model to include them in the vocabulary and retraining to compute their embeddings.

**Comparison of Results and Analysis Across Two Parts**

|  |  |  |
| --- | --- | --- |
| **Method** | **Spearman Correlation** | **Word Similarity Score (smart, intelligent)** |
| Transition Matrix Approach | 0.0617 | 0.1072 |
| Neural Network Model with Word Embeddings | 0.1496 | 0.4508 |
| Word2Vec Approach | 0.0073 | 0.9685 |

**Constrained Resources:**

**Trial No 1: Transition Matrix Approach**

Limited Performance in capturing semantic similarity between words as indicated by Spearman Correlation and word similarity.

**Trial No 2:** Neural Network Model with Word Embeddings

Here it demonstrated improved perfomance from Trial 1. The neural network was trained from scratch with corpus.txt (I Million Tokens) and capturing more semantic information and context.

**Unconstrained Resources:**

**Word2Vec:** This approach may have achieved a high similarity score for word pairs, but its performance measured by the spearman correlation coefficient, was lower compared to the neural network model.

b. **Phrase and Sentence Similarity :** In question (1) you would have come up with a method to get numerical/vector representation given a word. Now you have to come up with a mechanism to get representations for phrases and sentences. How do you aggregate individual word representations to get phrase or sentence embedding?

- You can use any pretrained static word embeddings like word2vec, GLOVE, FASTTEXT etc, or create your own.

- You can use popular tool/libraries (e.g nltk, Stanza, Spacy etc) to compute linguistic features (PoS, Constituency/Dependency Parse).

1. Phrase Similarity : Given a pair of phrases classify whether or not they are similar. Dataset can be found here. Dataset has train/dev/test splits. You have to report results on the test set, and use train/dev sets as needed.
2. Sentence Similarity : Given a pair of sentences, classify whether or not they are similar. Dataset can be found here. Dataset has train/dev/test split. You have to report results on the test set. , and use train/dev sets as needed.

**Solution:**

**In Common of Phrase Similarity and Sentence Similarity**

**Phrase similarity Dataset**: PiC/phrase\_similarity

**Sentence similarity Dataset**: google-research-datasets/paws (labeled\_final)

**Approaches for similarity predictions are:**

1. Static word embeddings using GloVe.
2. Contextual word embeddings using DistilBert

**Phrase Similarity**

**Trial 1: GloVe Embeddings**

First tokenization and removal of stop words using NLTK is been done. Then for embedding each word is converted to a 300-dimensional GloVe Vector. The phrase vector is obtain by averaging the word vectors. For modelling we used Logistic Regression classifier is trained on the concatenated phrase embeddings. For Inference similarity score is computed using cosine similarity between the phrase embeddings.

**Results:**

Validation Accuracy: 0.388

Test Accuracy: 0.3565

Similarity Score Inference:

“**newly formed camp**” and “**recently made encampment**”: 0.557

**Trial 2: DistilBERT Embeddings**

First using tokenization using DistilBERT tokenizer Then for embedding each phrase is converted to a 768-dimensional vector using DistilBERT. The phrase vector is obtain by averaging the word vectors. For modelling we used Logistic Regression classifier is trained on the concatenated phrase embeddings. For Inference similarity score is computed using cosine similarity between the phrase embeddings.

**Results:**

Validation Accuracy: 0.306

Test Accuracy: 0.296

Similarity Score Inference:

“**newly formed camp**” and “**recently made encampment**”: 0.4210

**Sentence Similarity**

**Trial 1: GloVe Embeddings**

First tokenization and removal of stop words using NLTK is been done. Then for embedding each word is converted to a 300-dimensional GloVe Vector. The phrase vector is obtain by averaging the word vectors. For modelling we used Logistic Regression classifier is trained on the concatenated phrase embeddings. For Inference similarity score is computed using cosine similarity between the phrase embeddings.

**Results:**

Validation Accuracy: 0.55325

Test Accuracy: 0.55175

Similarity Score Inference:

**Sentence 1:** "In Paris , in October 1560 , he secretly met the English ambassador , Nicolas Throckmorton , asking him for a passport to return to England through Scotland ."

**Sentence 2:** "In October 1560 , he secretly met with the English ambassador , Nicolas Throckmorton , in Paris , and asked him for a passport to return to Scotland through England ."

Similarity Score: 0.9966

**Trial 2: DistilBERT Embeddings**

First using tokenization using DistilBERT tokenizer Then for embedding each sentence is converted to a 768-dimensional vector using DistilBERT. The sentence vector is obtain by averaging the word vectors. For modelling we used Logistic Regression classifier is trained on the concatenated sentence embeddings. For Inference similarity score is computed using cosine similarity between the sentence embeddings.

**Results:**

Validation Accuracy: 0.5575

Test Accuracy: 0.56175

Similarity Score Inference:

**Sentence 1:** "In Paris , in October 1560 , he secretly met the English ambassador , Nicolas Throckmorton , asking him for a passport to return to England through Scotland ."

**Sentence 2:** "In October 1560 , he secretly met with the English ambassador , Nicolas Throckmorton , in Paris , and asked him for a passport to return to Scotland through England ."

Similarity Score: 0.3119

**Comparative Analysis Table**

| **Metric** | **GloVe (Phrase)** | **DistilBERT (Phrase)** | **GloVe (Sentence)** | **DistilBERT (Sentence)** |
| --- | --- | --- | --- | --- |
| Embedding Size | 300 | 768 | 300 | 768 |
| Validation Accuracy | 38.8% | 30.6% | 55.325% | 55.75% |
| Test Accuracy | 35.65% | 29.6% | 55.175% | 56.175% |
| Similarity Score Example | 0.557 | 0.4210 | 0.9966 | 0.3119 |
| Speed | Fast | Slower | Fast | Slower |
| Contextual Understanding | Low | High | Low | High |
| Computational Resources | Low | High | Low | High |

**Performance**:

* **Phrase Similarity**: GloVe embeddings performed better in terms of both validation and test accuracy compared to DistilBERT.
* **Sentence Similarity**: Both GloVe and DistilBert embeddings had similar performance, but DistilBERT showed slightly better test accuracy.

**Failures**:

* **GloVe** may fail to capture contextual semantics and polysemy.
* **DistilBERT** computationally intensive and may outfit if the dataset is small.

**Patterns**:

* **GloVe** embeddings struggle with phrases/sentences with polysemy.
* **DistilBERT** may overfit on the training data if not regularized properly.

**Speed and Resources:**

* **GloVe**: Faster and less resource-intensive, suitable for limited computational power.
* **DistilBERT**: Slower and requires more computational resources, but provide better computational power.

c. **BONUS TASK**:

1. Transformers are all the rage right now (backbone of most of the LLMs you might have used). Can you fine-tune a pre-trained transformer based models (BERT, Roberta, etc) to solve Phrase and Sentence Similarity Tasks described above? You are free to use any resource out there.
2. Can you prompt LLMs (ChatGPT, LLAMA) to solve the phrase and sentence similarity scores? Solve the task using
   * + commercial LLM APIs (ChatGPT, BARD etc);
     + open source LLMs/APIs (LLAMA, Mistral etc).

Try with zero and few shot settings. If querying LLMs is computational / commercially prohibitive, do it for only the test set / subset of test set. Analyze the results. Explain some analysis that you have done.

1. Compare all the approaches that you tried - static word embeddings, fine-tuned transformers, LLMs. What are the improvements you notice across the three settings?

**Solution:**

For the first part of the **BONUS TASK** I have already implemented Phrase similarity and Sentence Similarity with **DistilBert model** in the second part of the task. Also the comparative analysis with static embedding and pre-trained model is been done.

Now in the second part of the question we have computing phrase and sentence similarity scores using different models and short settings.   
  
**Phrase Similarity**

**1. Zero-Shot Setting**

**Comparison of the phrase similarity scores in the zero-shot setting**

| Phrase Pair | ChatGPT Score | Bard Score | all-mpnet-base-v2 Score |
| --- | --- | --- | --- |
| newly formed camp - recently made encampment | 0.80 | 0.75 | 0.7059 |
| one data - a particular statistic | 0.50 | 0.40 | 0.3324 |
| particular structure - specific edifice | 0.60 | 0.85 | 0.5267 |
| involved people - participating individuals | 0.85 | 0.90 | 0.6591 |

**Overall Consistency:**

* ChatGPT and Bard show relatively consistent scores for all phrase pairs, often within a 0.1-0.3 range difference.
* all-mpnet-base-v2 consistently scores lower than both ChatGPT and Bard for all pairs.

Variance in Scores:

* Bard shows the most variability in its scores, ranging from **0.40** to **0.90**.
* ChatGPT shows a moderate range of variability, from **0.50** to **0.85**.
* all-mpnet-base-v2 has the least variability, with scores ranging from **0.3324** to **0.7059**.

**2. Few Shot Setting**

**Comparison of the phrase similarity scores in the few-shot setting**

| Phrase Pair | True Similarity Score | ChatGPT Score | Bard Score | all-mpnet-base-v2 Score |
| --- | --- | --- | --- | --- |
| newly formed camp - recently made encampment | 0.8 | 0.8 | 0.90 | 0.7059 |
| one data - a particular statistic | 0.6 | 0.5 | 0.30 | 0.3324 |
| particular structure - specific edifice | 0.7 | 0.6 | 0.75 | 0.5267 |
| involved people - participating individuals | 0.9 | 0.8 | 0.80 | 0.6591 |

**Overall Consistency:**

* **ChatGPT** and **Bard** generally provided scores that were closer to the true similarity scores compared to the **all-mpnet-base-v2** model.
* Bard's scores tended to be slightly higher overall, while ChatGPT's scores were more balanced.

Variance in Scores:

* **Bard e**xhibits the most variability in its scores, ranging from **0.30** to **0.90.**
* **ChatGPT** shows a moderate range of variability, with scores ranging from **0.5** to **0.8**.
* **all-mpnet-base-v2** demonstrates the least variability, with scores ranging from **0.3324** to **0.7059**.

**Sentence Similarity**

**1. Zero-Shot Setting**

Comparison of the sentence similarity scores in the zero-shot setting

| Sentence Pair | ChatGPT Score | Bard Score | Sentence Transformers Score |
| --- | --- | --- | --- |
| The cat sat on the mat - The dog lay on the rug | 0.5 | 0.75 | 0.4844 |
| A bird chirped in the tree - The sun shone brightly in the sky | 0.2 | 0.00 | 0.2539 |
| She played the piano beautifully - He sang a song loudly | 0.6 | 0.25 | 0.2711 |
| The children ran and played in the park - The adults walked and talked in the garden | 0.5 | 0.30 | 0.1723 |

**Overall Consistency:**

* Both **ChatGPT** and **Bard** show consistent scores. They generally fall within a similar range.
* all-mpnet-base-v2 consistently scores lower than both ChatGPT and Bard for all pairs indicating a different approach to measuring similarity.

Variance in Scores:

* Bard scores range from **0.00** to **0.75**, indicating the most variability among models.
* **ChatGPT** shows a moderate range of variability, from **0.20** to **0.60**.
* **all-mpnet-base-v2** has the least variability, with scores ranging from **0.1723** to **0.4844** demonstrating the least variability and the most consistent performance across different sentence pairs compared to others.

**2. Few Shot Setting**

**Comparison of the sentence similarity scores in the few-shot setting**

| **Sentence Pair** | **True Similarity Score** | **ChatGPT (OpenAI)** | **Bard API** | **Sentence Transformers (**all-mpnet-base-v2**)** |
| --- | --- | --- | --- | --- |
| The newly formed camp is bustling with activity. - The recently made encampment is lively. | 0.8 | 0.75 | 0.80 | 0.7742 |
| One data point cannot represent the entire population. - A particular statistic may not reflect the overall trend. | 0.6 | 0.70 | 0.80 | 0.6612 |
| The particular structure of the building is unique. - The specific edifice stands out in the skyline. | 0.7 | 0.60 | 0.50 | 0.6238 |
| The involved people are dedicated to the cause. - The participating individuals show great commitment. | 0.9 | 0.80 | 0.85 | 0.8643 |

**Overall Consistency:**

* **ChatGPT** and **all-mpnet-base-v2** show relatively consistent performance.
* **Bard API** exhibits some variability in its scores.

Variance in Scores**:**

* **Bard API** has the narrowest range (**0.40**), indicating relatively consistent similarity scores across sentence pairs.
* **Sentence Transformers (all-mpnet-base-v2)** has the widest range (**0.2405**), suggesting more variability in the computed similarity scores.
* **ChatGPT (OpenAI)** falls between the other two methods in terms of range, indicating moderate variability in similarity scores.

**Overall performance of the methods**

* ChatGPT and Bard provide varying results, with ChatGPT generally producing slightly more consistent results across different examples.
* Sentence Transformers (all-mpnet-base-v2) provide similarity scores that are close to the true scores, demonstrating good performance without any tuning.
* Zero-shot settings show reasonable performance but generally improve when using few-shot settings with predefined examples.
* Static Word embedding have lack of context sensitivity and cannot capture nuances.
* Fine-tuned transformers can capture the contextual semantics however the require task-speific labeled data.
* LLMs offer state-of-the-artperformance in NLP understanding, excel in zero-shot and few-shot settings, and can provide dynamic text generation with minimal training data.

**d. Paper Reading Task :** BERTSCORE: EVALUATING TEXT GENERATION WITH BERT

Content provided in presentation

<https://www.canva.com/design/DAGHYWAD92I/_bg4KElaVJhkE5lUkuI16w/edit?utm_content=DAGHYWAD92I&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton>