Experiment 1

1. Using MURIL and BERT embeddings to analyse idiomatic expressions

The following explains the code and methodology in the "muril.ipynb" file

1. Sentence Embedding with MURIL - The code defines a function get model to load a pretrained MURIL model from TensorFlow Hub, which is designed for multilingual sentence embeddings. It utilizes BERT-style tokenization and generates embeddings for input sentences. .

2. Text Augmentation - The code uses the nlpaug library to perform synonym augmentation. It randomly selects a word from a given sentence (noun, verb, or adjective) and replaces it with a synonym. This augments the input sentence with a similar word.

3. Distance Calculation - The code calculates both Euclidean distance and cosine similarity/distance between pairs of sentences based on their embeddings. We provide functions (euclid dist() and cosine dist()) to compare the similarity/distance between sentences.

4. Conclusion - In summary, this code snippet showcases how we are able to compare cosine distance and euclidian distance between two sentences using embeddings generated by MURIL. We use nltk and wordnet to find words close in meaning for a given input word.. Using spacy we randomly select Adjectives,Nouns,Verbs from a given sentence which then using nlpaug we can replace either the entire sentence with an augmented form or replace a random word with a "similar" one. Then we compare the similarities to explore the level of contextual understanding of the MURIL embeddings

In the "bert.ipynb" we follow the same procedure, but here we use the more general BERT embeddings as our encoding scheme.

2. Utilized the NLTK library and spaCy to find synonyms for a given word and then select the most semantically similar synonym

1. Finding Synonyms with NLTK: NLTK's WordNet corpus is used to find synonyms for a given word. The find_closest_synonyms function retrieves synonyms by iterating through WordNet synsets and their lemmas. It removes duplicates and limits the number of synonyms to a specified count (num synonyms).

2. Semantic Similarity with spaCy: The code loads the spaCy medium English model (en_core_web_lg) to perform semantic similarity checks.

3. Selecting the Most Similar Synonym: The find_most_sim function calculates the semantic similarity between the input word and each synonym. It iterates through the synonyms, comparing their similarity scores to find the most similar one. The most similar synonym and its similarity score are returned.

4. Example Usage: The code demonstrates its functionality with the word "cake." It finds synonyms for "cake" using NLTK and removes underscores from any synonyms. It calculates semantic similarity scores for each synonym compared to "cake" using spaCy. The most similar synonym and its similarity score are printed. Output: In the example, the code finds that "patty" is the most semantically similar synonym to "cake" with a similarity score of approximately 0.539..

This function is utilized as an alternate form of augmentation of sentences for our previous embeddings based similarity experiments.

3. Utilising translation

By translating, back-translating, and comparing the original and back-translated text using Jaccard and cosine similarity metrics we analyse the utility of machine translation.

1. Library Imports: We start by importing necessary Python libraries including nltk for text processing, sklearn for similarity calculation, and googletrans for translation.
2. Data Preparation: We load a large text corpus from a file, split it into two halves,[with and without idioms] and subsample sentences for efficiency.
3. Translation and Back-Translation: We translate the first half of the text from its original language to French and Hindi and then back-translate it to English using Google Translate..
4. Jaccard Similarity: We calculate Jaccard similarities between original and back-translated sentences to measure word-set similarity.
5. Cosine Similarity: We compute cosine similarities between original and back-translated sentences using TF-IDF vectors, providing insights into semantic similarity.

NOTE : The textcorp.txt file is taken from the EPIE corpus
Text Corpus Link here

Conlusion

1. From our results we can draw the conclusion that the prebuilt BERT and MURIL embeddings cannot capture the meaning of idioms. Idioms often would seem nonsensical to these models and require more context to be understood.

2. Attempting similar experiments with a more complicated model like the google translate model, we hypothesized that to detect an idiom, when we translate it to another language (hindi and french) and then back to englsh, the back translated sentence will have little similarity to the original one assuming the model has internally understood the statement.

3. For example, "raining cats and dogs" translated to hindi and back to english is "Heavy Rain" which is the meaning. Now the similarity between these two english sentences is very low when we use syntactical similarity scores like the Jaccard Similarity measure which takes average number of similar tokens in both sentences.

4. Once more we noted that although this did yeild positive results for a few short idioms:

   - In hot water
   - Both go hand in hand
   - Don't sweat it
   - Beat around the bush
   - Call it a day
   - Easy as pie
   - Do not sweat it

   In a larger sentence with less commonly known idioms however we didnt see any consistent results, therefore proving that even extremely well known and complicated models do not have the capabilities to understand these types of sentences.