EXAMINING MULTILINGUAL NEWS ARTICLE SIMILARITY - PROJECT REPORT

Language Translation can be achieved by the Neural Machine Translation which fits a single model rather than a pipeline of fine tuned models and currently is achieving state of the art results.

1. What did you do and why is it interesting?

In this project we have explored machine translation for examining the similarities between the multilingual news articles.
Our objective is: Given a pair of news articles of the same language or different language, we examine whether they are conveying the same news/ story?
This task can also be categorised as a document level similarity task in the applied domain of news articles. The parameters used for calculating similarities are: Geolocation, Time, Shared Entities and Shared Narratives. We have chosen only these as the parameters as we are not interested in the style of the coverage, political spin, tone or any subjective design decision.
End result of the project is a score that has been awarded on a 4 point scale with 4 being most similar and 0 being the least similar.

It's fascinating because it allows us to cluster news articles and compare how different media outlets or regions cover the same stories.

2. What is your data ?

For the project we have a training data set of multiple languages. Primarily our data is news articles. Data set is presented in the form of a csv file that contains the links for the news covered by two different media entities. It also includes a score similarity of parameters like Geographic location, Entities, Time, Narrative, Overall, style and Tone. Also the data set has only one label that is the target label.
Training data set consists of 4964 pairs of news articles from the languages of english, spanish, german, polish, Turkish, arabic, french.
We articles in the data set provided are not accessible so we are able to access only 4606 pairs of articles.
For getting the test train data we have used the train test split command from the sklearn library. We split the data into an 80:20 division where 80% is the train data and the 20% is the test data.
Thereby getting the size of Training data as 3684 and,
The size of Testing data was 922.

And the pair of the news articles in the data sets are as follows.
English to English data sets : 1800
German to German data sets : 857
German to English datasets : 577
Spanish to Spanish datasets : 570
Turkish to Turkish datasets : 465
Polish to polish datasets : 349
Arabic to arabic datasets : 275
French to french datasets : 72
The dataset has

3. How do your models work?

Models we used in the project are :

1. Open source ecosystem Open MTpy and,
2. Logistic Regression.

Open source ecosystem OpenNMT-py architecture:

Neural Machine Translation (NMT) in general has the encoder decoder architecture , where the encoder segment learns the source language and the decoder learns about the target language. Both the encoders and decoders are a pre-trained neural network model (LSTM) in our case. After the model preprocesses the data and we create separate vocabularies for the source and the target language.
For changing textual information into a numeric form we use tf-idf which gives the words embeddings. We use tf-idf because it attached more importance to the words appearing more frequently.
We now feed this vector into the encoder neural network. We send the input in a sequential order like word after the word into the network. Note that we are not getting any output from the neural network except for the last one. Since we are using the LSTM variation of the RNN we are able to store the sequential input as LSTM can handle the sequential data. From the last layer of the encoder we get something called the thought vector which contains the context of the source language.

Then this thought vector is passed through the “bridge” which defines how the vector should pass from the encoder to the decoder. We used the default bridge, bridge copy which copies the states of the encoder. Note that the output sequence or the size of the decoder may not be the same as the encoder as the translation of language does not

mean that no of words in the document should be the same, it's the context that needs to be preserved.
Note that the mapping of the source language and the target language is many to many. In the Decoder we use special tokens <EOS> and <SOS> for marking the end of the string and the start of the string. These special tokens are passed as the hidden layers in the decoder and the thought vector is given as the input. Here the architecture changes from the encoder, we get the output from each hidden layer and then it is passed into the next layer. Pictorially the following diagram depicts the flow of the information. Then we use activation functions such as “relu” for introducing non-linearity in the output.
For our model we have used Adam optimization algorithm for the loss function optimization.

After we get the translated output, we use this output

We now get the output and use spearman’s coefficient on it which is then used to calculate the cosine similarity.
The output from the cosine similarity is then passed though logistic regression which yields us the overall similarity score.

Logistic Regression architecture:

We are considering and acknowledging the unknown words by replacing it by the <unk> token. Since we are using a pre-trained model we are not explicitly coding for handling the unknown words.

4. What are your results and conclusions?

Table 1: Pearson’s correlation matrix among different similarities calculated on Full processed text(train+text) and target human label. This matrix is for showing the underlying relationships only and no inferences from this are used while training, hence we used whole data for this. We see that all the similarities are negatively correlated with the human label and cos and arccos are highly correlated while BERT is quite different from cos/arccos.

	cos_similarity	arccos_similarity	use_similarity	bert_similarity	Human_Label
cos_similarity	1.000000	0.971511	0.678727	0.32685	-0.412418
arccos_similarity	0.971511	1.000000	0.659053	0.32609	-0.408130
use_similarity	0.678727	0.659053	1.000000	0.74229	-0.660861
bert_similarity	0.326850	0.326090	0.742290	1.00000	-0.496618
Human_Label	-0.412418	-0.408130	-0.660861	-0.496618	1.000000

Table 2: Baseline for the experiments is established by training LR and MLP models on combined unprocessed texts. The human labels for individual similarities (Geography, Entities, Time, Narrative,Style, Tone) will not be available in the final test set, hence we are not using them for training. The results are shown here just to show the correlations (Row 2, 3).

Feature Combination	MLP Pearson's correlation coefficient	MLP RMSE	LR Pearson' correlation coefficient	LR f1 score	LR Accuracy
Original Texts(baseline)	**0.3408**	**-0.0168**	**0.2984**	**45**	**47**
Individual Human Labeled Similarities	0.9358	0.8751	0.9103	77	77
Individual Similarities, Original Texts	0.3408	0.7895	0.91076	71	71

Table 3: The MLP and LR model results on the test set with different combinations of similarity features. We see the best performance when we use all four similarities as features. If we remove one of the cos/arccos similarity, the score only slightly reduces, which is consistent with the table 1 result that cos and arccos similarities are highly correlated. Only USE based similarity gives results comparable to the best model.

Feature Combination	MLP Pearson's correlation coefficient	MLP RMSE	LR Pearson' correlation coefficient	LR f1 score	LR Accuracy
BERT	0.4972	0.2455	0.399	32	44
cos, arccos, USE, BERT	**0.7077**	0.499	0.6748	49	54
arccos, USE, BERT	0.7042	0.4947	0.6752	49	55
USE, BERT	0.7001	0.4889	0.6750	**54**	55
arccos, BERT	0.6427	0.4127	0.5165	35	47
arccos, USE	0.7042	0.4944	0.6773	50	55
USE	0.6940	0.4809	**0.6802**	51	55
cos, BERT	0.6481	0.4175	0.5367	42	48
cos	0.3757	0.1393	0.3325	29	42
BERT Embeddings of both texts	0.2450	-0.462	0.224	35	38

5. Future experiments/ explorations and additions?

1. Instead of vectorizing the whole document, we can use topic modelling and use those topics as a feature.
2. At present, we are focusing on the context of the news covered by different media outlets, we can improve this by categorising the context further based on the tone of the style and cluster the similar style coverage (even from articles of different language) to get more insights about an event.
3. We could do more tests by eliminating stop words, which do not aid the model because they are abundant and do not contribute any unique information that may be used for classification and clustering.
4. For all circumstances, we may use cross validation to verify the model's performance on unknown data, which will provide us insights that will help us change our approach.
5. Prior to vectorization, we could do stemming (reducing words to their root words) because even if they are not stop words, they occur in a variety of

manifestations, do not provide any unique information for classification and grouping.

6. Using the approach of transfer learning, we can fine-tune the pre-trained models by reusing the elements of the pre-trained model in the new machine learning model.
7. Feature Engineering can be used to capture entities such as geographic locations, time frames, narratives, style, and tone in the data. The parameters can then be used to train models.
8. Only the BERT embedding model has been investigated; the similarity score may have been investigated as well. The results were unsatisfactory, and the training time was approaching exponential. It would be fascinating to investigate training on other embeddings in the future.

6. Citations:

1. Brownlee, J. (2019, August 7). *A gentle introduction to neural machine translation*. Machine Learning Mastery. Retrieved December 11, 2021, from https://machinelearningmastery.com/introduction-neural-machine-translation/.
2. Lanners, Q. (2019, June 7). *Neural machine translation*. Medium. Retrieved December 11, 2021, from https://towardsdatascience.com/neural-machine-translation-15ecf6b0b.
3. *Pretrained models¶*. Pretrained Models - Sentence-Transformers documentation. (n.d.). Retrieved December 11, 2021, from https://www.sbert.net/docs/pretrained_models.html.
4. *Scipy.stats.pearsonr¶*. scipy.stats.pearsonr - SciPy v1.7.1 Manual. (n.d.). Retrieved December 11, 2021, from https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html.
5. UKPLab. (n.d.). *UKPLAB/sentence-transformers: Multilingual sentence & image embeddings with bert*. GitHub. Retrieved December 11, 2021, from https://github.com/UKPLab/sentence-transformers.
6. Tensorflow hub. (n.d.). Retrieved December 11, 2021, from https://tfhub.dev/google/universal-sentence-encoder-multilingual/3.
7. Yang, Y. (n.d.). *Learning semantic textual similarity from conversations*. ACL Anthology. Retrieved December 11, 2021, from https://aclanthology.org/W18-3022/.
8. ryuzakinhoryuzakinho 22322 silver badges66 bronze badges, Nominal AnimalNominal Animal 8, & SomosSomos 28.5k22 gold badges2626 silver badges6060 bronze badges. (1966, September 1). *Cosine similarity vs angular*

distance. Mathematics Stack Exchange. Retrieved December 11, 2021, from https://math.stackexchange.com/questions/2874940/cosine-similarity-vs-angular-distance.
9. *Competition*. CodaLab. (n.d.). Retrieved December 11, 2021, from https://competitions.codalab.org/competitions/33835#learn_the_details-overview.
10. OpenNMT. (n.d.). Retrieved December 11, 2021, from https://opennmt.net/Models-py/.
11. *Pretrained models¶*. Pretrained Models - Sentence-Transformers documentation. (n.d.). Retrieved December 12, 2021, from https://www.sbert.net/docs/pretrained_models.html.
12. *Cosine distance cosine similarity angular cosine distance angular cosine similarity*. COSINE DISTANCE, COSINE SIMILARITY, ANGULAR COSINE DISTANCE, ANGULAR COSINE SIMILARITY. (n.d.). Retrieved December 12, 2021, from https://www.itl.nist.gov/div898/software/dataplot/refman2/auxillar/cosdist.htm.
13. Varun. (2020, December 3). *Calculating document similarities using Bert and other models*. Medium. Retrieved December 12, 2021, from https://towardsdatascience.com/calculating-document-similarities-using-bert-and-other-models-b2c1a29c9630.