EXAMINING MULTILINGUAL NEWS ARTICLE SIMILARITY PROJECT REPORT

About the project

We live in a globalized world in which speakers of many different languages tend to interact with each other. Various financial events can be taken from news sources in different languages and can be discussed by these speakers. News articles consists of long texts. We need a fast and reliable method to calculate the similarity be-tween these documents. The pairs of articles are considered to be similar if the topic discussed is the same and they have the same meaning. We are examining various methods that will automati-cally help in the calculation of the similarity scores.

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. 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 be categorised as a document level similarity task in the applied domain of news articles. End result of the project is a score that has been awarded on a 4 point scale with 4 being most similar and 1 being the least similar. It's fascinating because it al-lows us to cluster news articles and compare how different media outlets or regions cover the same stories. The performance of the models is evaluated using Pearson's correlation coefficient.

About the Data

For the project we have a training data set of news articles in multiple languages. 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, Narra-tive, style and Tone. Also the data set has one target label "Overall". Training data set consists of 4964 pairs of news articles from the languages of english, spanish, german, polish, Turkish, arabic, french. Few 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 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. The gold labels in the dataset are calculated by taking an average scores given by human annotators based on different criteri-ons. There can be 1-8 human labels for a pair of articles.[9]

And the language pairs 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

Working of the Models

Pearsons correlation coefficient

<Please refer the formula diagram given at the end of the document>

This coefficient is mainly used to measure the similarity be-tween two documents.The first document's vector is being taken as x and the second document's vector is taken as y.

The Pearson correlation coefficient r (float) returns a value between -1 and 1 . The value 0 implies no correlation. If the values of correlation are +1 and -1, then this indicates an exact linear relationship.[4]

BERT- Bidirectional encoder representation from trans-formers. BERT uses an attention model which helps to learn embeddings for words. The representation of training text is using three sets of embeddings which are token , segment and position. We are using a pre-trained BERT model (multi-qa-MiniLM-L6-cos-v1) to embed our documents.[3, 5]

USE - Universal Sentence Encoder

It encodes text into high dimensional vectors. The inputs are variable length text and the output is a 512 dimensional vec-tor. We will be using the pre-trained USE which is available in Tensorflow-hub. It is being trained with Convolutional Neural Network and it covers a total of 16 languages. USE makes it easier to get the sentence level embeddings which will then be helpful for computing sentence level meaning similarity. For similarity measure, we use the cosine similarity on these USE embeddings. Higher score indicates greater similarity between the documents.[6]

Cosine Similarity
<Please refer the formula diagram given at the end of the document>

Cosine similarity is used to identify the similarity between two documents by calculating the cosine of the angle between the two vectors which are derived from the docu-ments. This is a measure of the correlation between the sen-tence embeddings which are represented by the vectors. Smaller angles mean that the documents are more similar and larger angles indicate that the documents are different.
We will be calculating the dot product of the document vec-tors. It makes nearly parallel vectors similar. The dot prod-uct for similar vectors is 1 and for dissimilar vectors is 0. So the value is between 0 and 1.

arccos Similarity
Yang et. al discuss that arccos similarity performs better than cosine similarity in capturing semantic meaning.[7]
arccos similarity is similarity based on angular distance between the two vectors. It is used for converting the cosine similarity scores into angular distances which obey the triangle inequality.
<Please refer the formula diagram from the end of the document >

TF-IDF - Term Frequency Inverse Document Frequency
<Please refer the formula diagram from the end of the document >

Using TF-IDF, we are trying to find how important a particular word is to a document. Term Frequency-If a word oc-curs more frequently in the document, it is very important to the document. Inverse Document Frequency- This value will be very low if the words appear in many documents. We are using TfidfVectorizer from Scikit-Learn for vector-izing the documents. This algorithm will transform the text documents into a vector of numbers. It compares the number of times a word appears in a document with the number of documents the word appears in. We think that TF-IDF will better capture the semantic meaning than one hot encoding because it takes the term frequencies into context.

Open source ecosystem OpenNMT-py architecture[10] Neural Machine Translation (NMT) in general has the en-coder 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 vocabu-laries for the source and the target language. For changing textual information into a numeric form we use vectoriza-tion(tf-idf, USE, BERT) which gives the words embeddings. The model feeds this vector into the encoder neural network and sends the input in a sequential order like word after the word into the network. Note that we do not get 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 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. The output sequence or the size of the de-coder may not be the same as the encoder as the translation of language does not mean that number of words in the doc-ument should be the same, it's the context that needs to be preserved.
The mapping of the source language and the target language is many to many. The pre-trained model we are using han-dles the unknown words by replacing it by the <unk> token and we are not explicitly coding for handling the unknown words. 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.
The baseline, we run logistic regression (LR) and multi-layer preceptron regressor (MLP) on combined and processed text data. The output from the translation is then con-verted into embeddings, then we calculate similarity. For similarity calcualion we consider 4 types of methods namely: cosine similarity, arccosine, universal sentence encoder and BERT. These similarities are then used as features to train LR and MLP regression models which gives us the predictions. Since the target label is a continuous valued number, we can not have accuracy as the evaluation crite-rion as it is a regression problem. We use the predictions and test set to calculate the pearson’s correlation co-efficient for evaluating the performance.

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 la-bel and cos and arccos are highly correlated while BERT is quite different from cos/arccos.

<Please refer table 01 from the images given at the end of the document>

Table 2: Baseline for the experiments is established by train-ing 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).

<Please refer table 02 from the end of the document>

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 fea-tures. If we remove one of the cos/arccos similarity, the score only slightly reduces, which is consistent with the ta-ble 1 result that cos and arccos similarities are highly cor-related. Only USE based similarity gives results comparable to the best model.

<Please refer table 03 from the end of the document>

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 explored removing stopwords in german and english articles for cosine similarity, but the results did not improve, so they are not included in the results table 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. We tried cross validation for LR and the results were same as the default configuration of sklearn. We can do further experiments on cross validation to optimally select the hyperparameters 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 with the available training data.

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. As it was clear from the table 2 results that we get good performance by using these features.

8. We can also explore traing on the actual embedding instead of the similarity. We did this for BERT and the results were unsatisfactory, and the training time was approaching very high. It would be fascinating to investigate training on other embeddings in the future.

9. We can explore adding support to more languages for the translation and then similarity computation.

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-Trans-formers documentation. (n.d.). Retrieved December 12, 2021, from https://www.sbert.net/docs/pretrained_mod-els.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/gener-ated/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, & Somos-Somos 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. 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/soft-ware/dataplot/refman2/auxillar/cosdist.htm.

12. Varun. (2020, December 3). Calculating document sim-ilarities using Bert and other models. Medium. Retrieved December 12, 2021, from https://towardsdatascience.com/calculating-document-similarities-using-bert-and-other-models-b2c1a29c9630.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

Pearson’s corelation formula

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

$$\text{angular cosine distance} = \frac{1/\text{cosine similarity}}{\pi}$$

$$\text{angular cosine similarty} = 1 - \text{angular cosine distance}$$

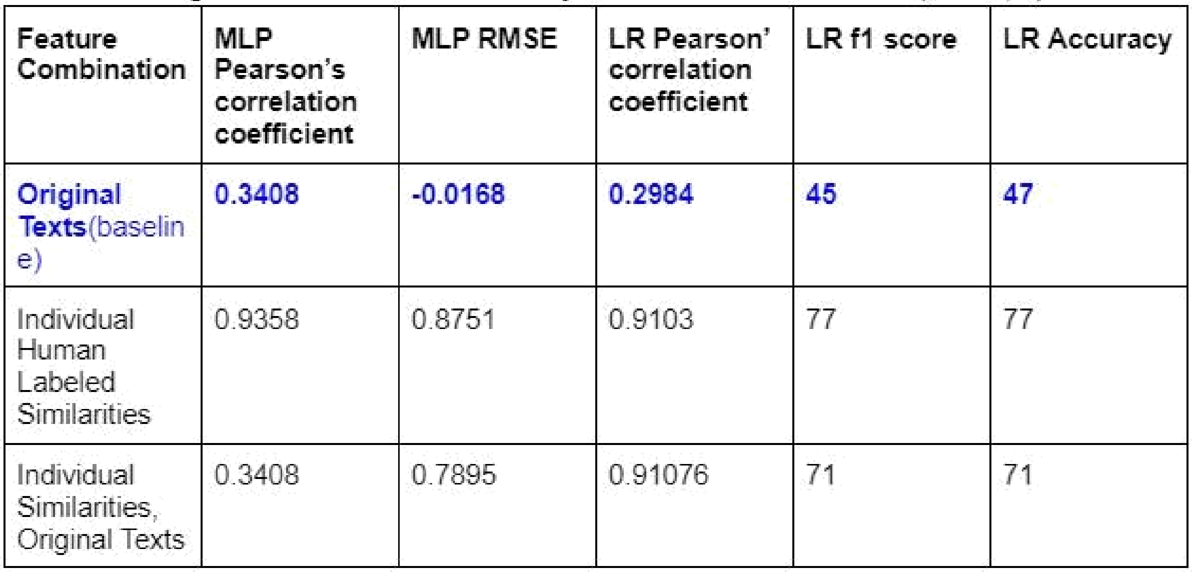
TF = (Number of time the word occurs in the text) / (Total number of words in text)

IDF = (Total number of documents / Number of documents with word t in it)

TF-IDF = TF * IDF

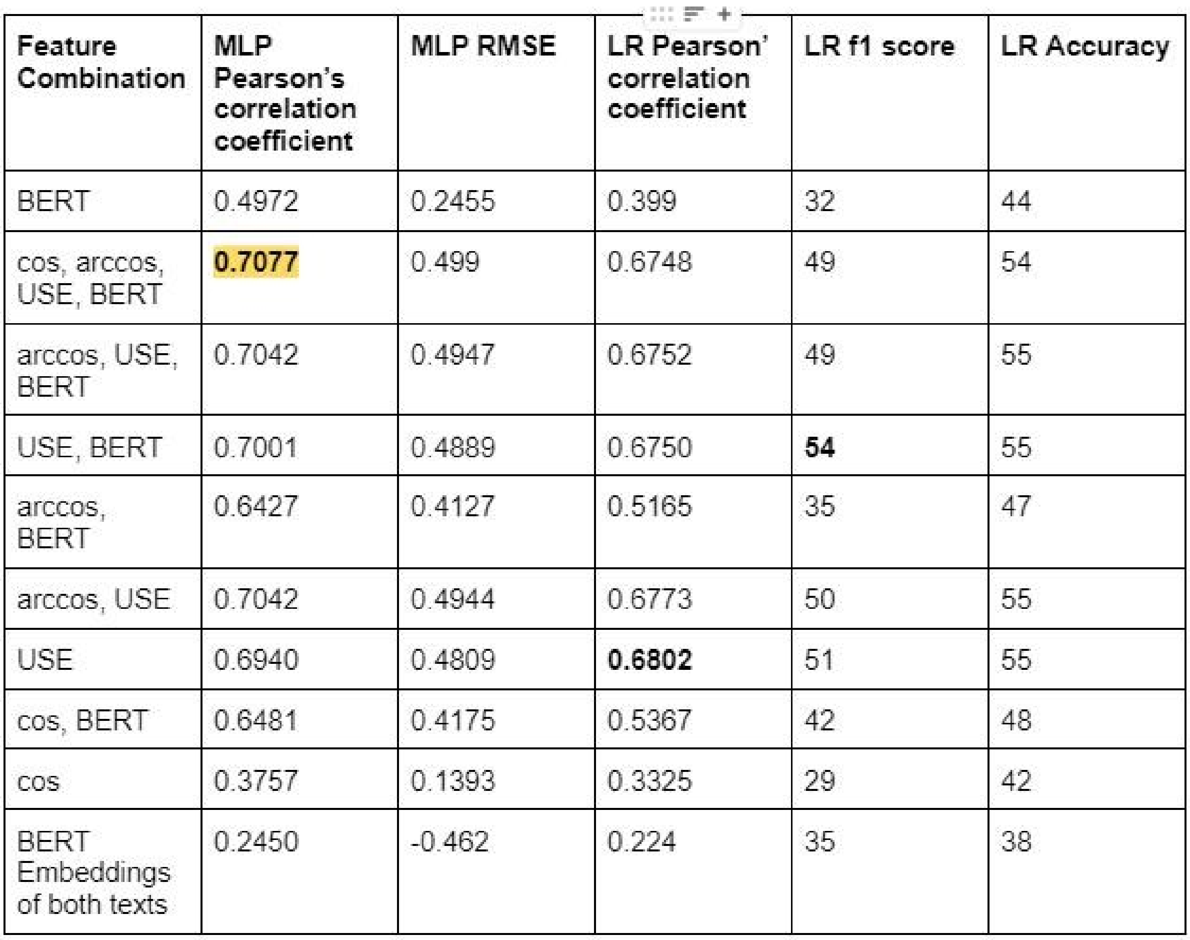
	cos_similarity	**arccos_similarity**	**use_similarity**	**bert_similarity**	**Human_Label**
cos_similarity	1.000000	0.971511	0.678727	0.32685	-0.412418
arccos_similarity	0.971511	1.000000	0.659053	0.32609	-0.408130
use_similarity	0.678727	0.659053	1.000000	0.74229	-0.660861
bert_similarity	0.326850	0.326090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

Table 01



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 02



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

Table 03

[END OF THE DOCUMENT]