CO490 Natural Language Processing Coursework

Nithya Krishnan
CID: 01668396
nk1419@ic.ac.uk

Wan Qi Ang
CID: 01668459
wqa19@ic.ac.uk

Abstract

The aim of the task is to develop a a variety of regression models to predict the quality and accuracy of a machine translated sentence given a pair of source (English) and machine translated (German) sentences. To train the models, several pre-processing techniques were applied on the raw dataset of 8000 sentence pairs and multiple methods of embedding the sentence pairs to vector were also constructed to prepare them as input to the models. The embedding methods are then combined with different regressions models to form a model that can predict the quality of a machine translated sentence. Each model is analysed based on the results from the test and challenges such as a biased training dataset are discussed to provide an insight into the test results.

1 Text Pre-processing

On an initial scan of both English and German corpus, it was identified that most sentences were formal sentences with semantic structure. In order to clean our raw data and achieve a standardised format across the samples, we conducted simple text pre-processing tasks before proceeding with tokenization of the sentences. (Kannan and Gurusamy) Additionally, for certain models where Part-of-Speech (POS) tags or Named-Entity-Recognition (NER) tags were required, the tags were obtained from the complete English and German sentences before the above listed pre-processing is done.

1.1 Conversion to lowercase

Upon inspecting the dataset, particular samples contain capitalised words for person name so we have to convert all words to lowercase for standardisation. To do this, we make use of the function lower() to convert the corpus to lowercase.

1.2 Splitting connected words in the corpus

There exits words connected with '-' or '/' in both corpuses, especially in the German sentences. These connected words will affect further embedding as they will be considered unknown words and will be less frequent in the corpus. Hence, these words are split using the python package re.

1.3 Removal of Stop Words

The removal of stop words is seen as important as it allows us to lessen our processing time by removing words that do not add value. (K. and Saini, 2016) To remove the stop words, we used NLTK's list of stopwords for both English and German.

1.4 Removal of Punctuation and symbols

The removal of punctuation is done by using the re.sub function to replace the punctuation with white spaces. Additionally, symbols and end-of-sentence and tab symbols are removed.

1.5 Part-of-Speech Tags

The Part-of-Speech (POS) tags for each sentence in both languages can be identified using the package Spacy. These POS tags could provide information about the grammar on each sentence.The POS tags are then kept in a list to be further used for pre-processing.

1.6 Named-Entity-Recognition Tags

The Named-Entity-Relation (NER) tags for each sentence in both languages can be identified using the package Spacy. These NER tags could identify words that cannot be translated as the word is an entity. The NER tags are then kept in a list to be further used for pre-processing.

2 Embedding

Each regression model incorporated one or more of the following word or tag embedding methods.

2.1 Bag-of-Words

Bag-of-Words (BoW) model is a simple way to extract features from text and turns the text into fixed-length vectors by counting the number of times each word appears. To implement the BoW model ,we needed 2 main things - the vocabulary of known words and a way to measure the presence of known words. (Brownlee, 2019)

Since the BoW model do not take the order of words into account, we can simply count the number of times each known words appear in our text and convert the sentences into a fixed length vector with the size of our vocabulary containing the count of each word in the sentence.

2.1.1 Getting the vocabulary of known words

To get the vocabulary of known words, we first do text pre-processing as mentioned in 1 and then tokenized the training set corpus to get the word tokens. By iterating through the tokenized corpus, unique words are added into the vocabulary list. To minimise the sparsity of data in the vectorized text later on, only the most frequent 600 words are used in the final vocabulary.

2.1.2 Translating sentences into vectors

With the vocabulary of known words ready, each sentence is then vectorized by counting how many times each word in the vocabulary appears in each sentence.

2.2 BERT

BERT is a well-known pre-trained word-embedding approach. (Horev, 2018) BERT builds open the concept of transformers - models that process words in relation to all other words in a sentence - to obtain contextualised word vectors.

We passed the each entire lower-cased sentence into the BERT model to obtain accurate word vectors with the nuances of context. The vectors corresponding to numbers, punctuation, symbols or language-specific stop words were then removed and the Norm of the remaining word vectors are added to the training corpus. The training corpus were then presented either as the vector difference between each English sentence and the German translation or the the English sentence's vector concatenated with that of the German translation.

We experimented with different types of BERT models which are identified below.

2.2.1 BERT English and BERT German

The `en_trf_distilbertbaseuncased_lg` BERT model was loaded onto the package Spacey to convert English words in a sentence to its respective word tokens. The same was done for German words using the `de_trf_bertbasecased_lg` BERT model. Both packages are able to identify the NER and POS tags for each word.

2.2.2 BERT Multilingual

`distiluse-base-multilingual-cased` is a new BERT multilingual model. The same pre-trained model can be used to obtain word vectors for both the English and machine translated German sentences. (Pires and Schlinger, 2019) Thee package, however, is unable to identify the NER and POS tags for each word.

2.3 Keras Tokenizer

The Keras Tokenizer simply splits sentences into words, filters out punctuation and converts text to lower case before encoding both the English and German sentences.

2.4 POS Tag Embedding

A list of POS tags in each sentence for both languages are obtained during text pre-processing can be embedded into the training corpus as a vector. The length of the vector is determined by the number of unique POS tags for each language. Thereafter, the counts of each type of POS tag per sentence was added to the vector. The final POS tag vectors were concatenated to the training corpus.

2.5 NER Tag Embedding

A list of NER tags in each sentence for both languages are obtained during text pre-processing can be used in two ways. The tags can be used to identify the word vectors of named entity words and remove the word vector from the sentence vector. The tags can also be used to replace the word vector of a named entity with a customised word vector that represents the type of entity.

3 Model Design

The training dataset was optionally scaled if it improved model performance or if Principal Component Analysis (PCA) was done. PCA was used in the cases where the training dataset had a large number of features (more than 500) to reduce the number of features while effectively maintaining

95% of the variance within the dataset. All regression models are built upon Sklearn or Keras frameworks. The best hyperparameters of each model and optimizers were identified through a package called talos. talos performs random optimisation with probabilistic reduction to try a subset of all possible permutations to achieve the best results. Additionally, we performed k-cross validation on all models with 8 splits to evaluate the models against a limited data sample and reduce model bias.

3.1 Support Vector Machine

The Support Vector Machine (SVM) found in the sklearn package was used with different kernels, mainly the linear, sigmoid and RBF kernel. The SVM was chosen due to the model's ability to work with small samples but high-dimensional data well. (Stecanella, 2019)

3.2 Random Forest Regressor

The Random Forest Regressor (RFR) found in the sklearn package was used with `n_estimators` parameter optimised. This ensemble model is able to offer efficient estimates of the test error without requiring to do cross-validation. (Krishni, 2019)

3.3 Extreme Gradient Boosting

The Extreme Gradient Boosting (XGBoost) model was used with multiple parameters such as `max_depth`, `reg_alpha`, `n_estimators` and `gamma` optimised. This ensemble model is able to achieve more accurate model performance with greater computational speed. (Unknown, 2019)

3.4 Multilayer Perceptron

The Multilayer Perceptron (MLP) model was built using Keras layers, optimizers and loss functions. The model's first layer is an Embedding layer. The second layer is an LSTM layer which is a recurrent neural network that attempts to model sequence based behaviour in this case. Thereafter, a deep model compromising of alternating Dense and Dropout layers with a few were used (Pathmind). Parameters such as the `hidden_size`, `dropout_value`, `activation`, `optimiser` and `learning_rate` were optimised.

4 Performance of Models

The below models are built using different combinations of word embedding and regression models and are listed in descending order of Pearson score.

	Model	Pearson
1	Multilingual BERT with XGBoost	0.11712
2	BERT (with Named Entity kept) with RFR	0.08473
3	Keras Tokenization with MLP	0.0554
4	BERT (with Named Entity removed and POStagged) with Rbf SVM	0.02795
5	BERT (with Named Entity removed) with-Linear SVM	0.01096
6	Bag-of-Words with Linear SVM	-0.0096

Figure 1: Pearson scores for each model

4.1 Multilingual BERT with XGBoost

A BERT Multilingual word embedding method without the adding or removal of tags with the XGBoost algorithm as a regression model was deployed. As each sentence embedding has 512 features, the vector difference between the English and German vectors was obtained and PCA was performed on the training dataset to reduce the number of features to 208.

A reason the model was successful is possibly due to the fact that words across all languages which are similar semantically are mapped to the same word vector - making it easier to score the quality of the machine translation. However, a possible downside to this method is English words within the German corpus are not penalised and can be mapped effectively. This can be seen in the translation of the sentence - "Illustrated Encyclopedia of Woodworking Handtools, Instruments Devices." in the training corpus which yields a predicted value of 0.51 while the actual value is -6.77.

4.2 BERT (with Named Entity kept) with RFR

Separate language BERT models for word embedding was used, with NER tags used to replace entities with a common vector. The English and German vectors were concatenated together and input into the RFR model to predict the translation score.

Compared to other models with entity word vectors removed, this model performs better. By replacing a word vector of Norm zero with a non-zero value that corresponds to a specific type of entity, the RFR is fed more useful information than a sparse vector with multiple zeroes.

4.3 Keras Tokenization with MLP

A common Keras Tokenizer was used to convert both English and German sentences into it's embeddings and the 2 datasets was concatenated in the third dimension. The resulting dataset was input into the MLP model.

Although the model had an average performance, we noted that the model requires minimal preprocessing and training (1 epoch). Moreover, a deeper network with fewer neurons per layer performed better.

4.4 BERT (with Named Entity removed and POS tagged) with RBF SVM

Pre-trained word embeddings were received from separate English and German BERT models, with NER tags used to remove entities from the sentence vector. The vector difference between the English and German vectors was used to build the training dataset. Additionally, the POS tags from both English and German sentences were concatenated together along with the training dataset and fit into the RBF SVM model.

While the POS tags are available to provide some information about the grammatical structure of the sentence (Ma and Huang, 2011), the impact of the additional information on the score is limited. This could possibly be because the word embeddings in BERT are already contextualised to include the meaning of the word and POS with respect to the sentence.

4.5 BERT (with Named Entity removed) with Linear SVM

Separate language BERT models for word embedding was used, with NER tags used to remove entities from the sentence vector. The English and German vectors were concatenated together to form the training dataset which was input into the Linear SVM model to predict the translation score.

The model performed performed poorly as compared to the model with the entities replaced with a common vector. This could be as there are sentences in the corpus such as which are mainly made up of entities. When these sentences are converted to their word vectors with the vectors representing the entity removed and padded to a constant length, the sentence vector becomes a sparse vector and does not provide any useful information while training the model. (Finkel) Hence, replacing entities with a common vector representing the type of entity is preferred.

4.6 Bag-of-Words with Linear SVM

The Bag-of-Words (BoW) model is used to convert the sentences into vectors of word count for both English and German text. The sentence vector is then concatenated together as the processed training data and then fed in as input to the linear SVM model to train it and predict the scores for the test set. Comparing to all the other models derived, this model performed the worst. The low score could be attributed to the pre-processing method used as BoW do not take into account the context and sequence of words in the sentences. As sentence semantics matter in language translation, BoW model is expected to perform badly in this area. In addition, it is possible that the sentence vectors would consist of large number of zeros given the large vocabulary size. In this case, the sparse representations are harder to model since there is a limited amount of information that can be harnessed in a huge representational space.

4.6.1 Dealing with large vocabulary

Due to the large dataset, one of the challenges faced during the BoW approach was to seful features of the text without using high dimensional vectors. Given the limitations of computational power and time, we chose to select the 400 most frequent words in our vocabulary as our BoW. As a result, this might have largely decreased the accuracy of our model since the sentence vectors might have already lost a chunk of useful information.

5 Overall Insights/Challenges

In conclusion, we identified that maintaining sentence semantics in our word embeddings were crucial in predicting the translation score. Additionally, using NER tags to replace entity word vectors with a common vector boosts performance. The choice of regression model varies with the type of pre-processing and no one model fits all.

The majority of scores for the training dataset ranged between 0.1 and 0.2 as the scores were z normalised. This causes most models to be biased as they are able to reduce the mean-squared-error (loss), by outputing all values ranging from 0.1 and 0.2. Additionally, the scores for the test dataset did not have the same probability distribution as the training dataset hence it was difficult to obtain a good Pearson coefficient for all models.

References

Jason Brownlee. 2019. A gentle introduction to the bag-of-words model.

Jenny Rose Finkel. Joint parsing and named entity recognition. Master's thesis.

Rani Horev. 2018. Bert explained: State of the art language model for nlp.

Jaideepsinh K. and Jatinderkumar Saini. 2016. Stopword removal algorithm and its implementation for sanskrit language. *International Journal of Computer Applications*, 150:15–17.

S Kannan and Vairaprakash Gurusamy. *Preprocessing Techniques for Text Mining*.

Krishni. 2019. A beginners guide to random forest regression.

Jianjun Ma and Degen Huang. 2011. Pos tagging of english particles for machine translation.

Pathmind. *A Beginner's Guide to Multilayer Perceptrons (MLP)*.

Telmo Pires and Eva Schlinger. 2019. https://arxiv.org/pdf/1906.01502.pdf.

Bruno Stecanella. 2019. An introduction to support vector machines (svm).

Guest Unknown. 2019. Understanding the math behind the xgboost algorithm.

A Appendices

The source code of all models built in this paper can be found in the following GitHub repository: NLP Machine Translation.