Classification on Parallel Corpora in Translation Relation Using Deep Learning Techniques

Shivendra Bhardwaj

***Abstract*—This paper present the classification technique used on the bilingual parallel corpora. Most of the work in this paper is are inspired by the latest machine translation technique being used around the globe. The paper shows that how can we efficiently learn the given corpus of two different language and combine them to understand the pattern. The baseline approach describes novel method which are used for classification, further introduced Deep Learning approach to understand the context of**
both corpus and learn the pattern. The Deep Learning approach out performed the baseline by 8% (classification error), the matrix used to measure the model performance are F-Score and Confusion matrix.

***Index Terms*—Classification, Machine Translation, Bi-LSTM, Convolution Net, Word Embedding**

I. Introduction

In the current world of Artificial Intelligence, machine translation is one of the novel problem, and it's development played an important role in our day today life as well. In this problem there are two parallel corpora, one in English language and second in French language, there is a binary class variable as well to label each combination of parallel translation as true or false. The main task here is to train model on the above data and learn the true translation to classify the unlabeled data set. The state of art technique used in learning such patterns are Bi-Directional Recurrent Neural Network (Bi-LSTM) and Convolutional neural network (CNN) which are applied in this paper and the result have been compared with the vanilla flavored classification model (baseline).

II. Corpus Description

The corpus used in this work was from domain European parliament data set, one in English and another in French with labels mentioned whether is correct translation or not. In this work, the ground truth is that the data labeled as '1' is correct translation and there are no external translation data which are used to train the model. The size of the training data is 1 million with 50% as true translation and rest are negative samples, the data set looks balanced in some sense that the both class are of equal
size, but this could also lead to over fitting. The goal of the exercise is to determine the wrong translation, and there are very fewer data for negative sample for this case, the reference paper [1], Extracting Parallel Sentence using RNN [1], the negative sample used was 10 per 1 positive translation. The below table (Table 1) gives a bird's eye view of the corpus used, in the model which will also help us in determining the parameters in the models. There are some surprising facts' comes out here, one is that the size of vocabulary in English is significantly lower than that of French, which is also visible in the maximum size and average size of sentence in the corpus.

TABLE I
Exploratory Analysis
after cleaning (removal of punctuation, number etc)

Language	Corpus Size	Vocab Size	Smallest Sentence Size	Largest Sentence Size	Average Length
English	1 M	80,742	2	344	25.73
French	1 M	105,810	2	774	28.47

We can presume the fact that translation in French takes 29% more number of words to describe the same fact as English, but this assumption could be wrong since we are looking at both negative and positive sample, so next we are exploring the sentence length by separating them into two class of positive and negative. In Figure 1, describe the difference between the length of each parallel sentence in both language (Len(English — Len(French)), where the first 500,000 are true translation and last 500,000 are from negative sample. It is very clear that the true translation are far less than the negative sample, this is very interesting fact since, the distribution in true sample is range between o-10 with few outliers, but in negative sample it goes way above 40 and with few outliers more than 800.

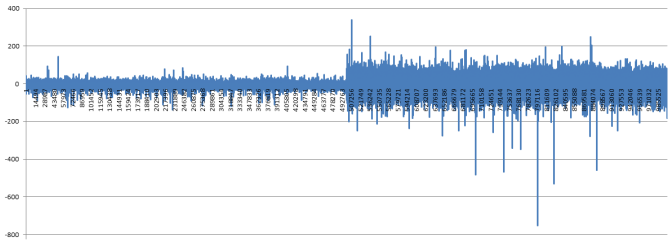

Fig. 1. Difference between the #words of each parallel corpora

We have to disprove our assumption that French sentences have more words when compared to its parallel sentence in English, from above it is clear that for true translation the difference between the length sentences are very less and consistent, which tells us that it could be used as very important classifier to our model.

Shivendra Bhardwaj (e-mail: shivendra.bhardwaj@umontreal.ca)

III. Multilingual Sentence Embedding

From the base reference paper [1], we learned that we need to use word embedding, which is much better than Bag-of-Words or any similar techniques. Reference [2], Word embedding provide a dense representation of words and their relative meanings. They are an improvement over sparse representations used in simpler bag of word model representations.

There could be multiple methods to solve the problem we have discussed till now, but we are going to solve this paper based on the perspective of a Machine Translation, so we need to build our language model based on the two corpus we have. We have used some external data uplift our learning here, which is FastText data from Facebook.

What is FastText data ? FastText is a word embedding not unlike Word2Vec or GloVe, but the cool thing is that each word vector is based on sub-word character n-grams. That means that even for previously unseen words (e.g. due to typos), the model can make an educated guess towards its meaning, reference [3].

One of the most important reason we have used this external embedding to our model is, learning from 1 million parallel corpora with 80,742 English vocab and 105,810 French vocab, is difficult and in this problem we need strong relation between the words, which is hard if we use only the corpus we have. Hence, we learn the wights from the FastText word embedding which has 300 dimensions with 2,519,396 tokens for our English and 1,152,438 tokens for French corpus.

IV. Baseline Model

All the experiments were performed using keras library and embedding weights learned from FastText. This is the very basic vanilla flavored model, with just the two word embedding layers with the learned weights from FastText vectors space and the labels to train the model. The reason why we tried this model is to test how capable is the corpus alone to classify without any other classifier.

Architecture :

- Input layer contains embedding with prelearned weights, the maximum number of words per sentence in both the embedding is restricted to 40. The number is chosen since the average number of words is 25 and 28 for English and French respectively and from the difference graph we can see the difference is something more than 40 visually, which tells that those sentences are quite long in either of the language. Hence, if we keep maximum number of words per sentence for whole corpus close to 40, which can capture more of true translation than that of negative sample. This number is consistent throughout all the models, and it also shows that if we increase this number it also increases the noise in the data set.
- The two embedding layers are merged together to push into the layer, the layers are averaged rather than concatenate, since concatenate will lead to huge matrix and hard for model to learn the space with the size of data we have. The Average function make more sense here since we believe both language has similar vector space. We can also use Subtract function, but the negative sample is quite less compared to general machine translation problems and the reference paper [1] used 10 negative sample per positive sample.
- In this model, there are one hidden layers with 64 neurons, since our input layer is of 40 words, just to have hidden layer to be close to double the size of input layer.
- Sigmoid activation function is used in the output layer, for binary classification. The main reason why we use sigmoid function is because it exists between (0 to 1).Therefore, it is especially used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1**,** sigmoid is the right choice. $\phi = 1/(1 + e^{-z})$ Source: Wikipedia

Results :

- Here the accuracy is 71.17%, which is very good from a single layer model, which mean the word embedding has good capability to predict classes.
- The recall is 56% and precision is 83% and F1 score is 69%, we have calculated these matrices to compare our other classification models since just accuracy doesn't make whole sense in this case of problem.
- There is one problem with this one layer model is that the training accuracy is way higher (89%) compared to the test, which is a clear sign of over fitting.

TABLE II
Confusion Matrix — Baseline model with one hidden layers

	Predicted label '0'	Predicted label '1'
Actual '0'	86,322	13,678
Actual '1'	43,977	56,023

V. Improving Baseline model

A.

*1) **Model 1 — New Features and Improved layers:**:* There are multiple ways we can go from here, but for now we will introduce more features for model to learn here, from the past results we can see that 43,977 true translation data points we miss classified to negative samples, which is a big issue, (i.e recall = 0.56). Although we are able to achieve 71% but, the model is good at classifying wrong translation (label '0') more efficiently than true translation.

There is always a trade off between these matrices, so for this work we want to improve on our true translation (label '1'), rather than wrong translation (label '0'), since machine translation is all about giving the right translation. Hence, from here on we are going to focus on our Recall rather than Precision.

Improvement in feature space:

- There is a clear indication that we need more efficient features, one very important feature we saw above in Fig 1, the difference between the length of each sentences in parallel corpora, we will concatenate it to our hidden layer output of our model.
- There is another feature we can add is 'lexique.en-fr', which is basically a dictionary for English to French

word, and it could be quite useful. This dictionary will help us to convert the *English token into French*, so for whole English corpus we convert all possible words available in dictionary to French. We create a new embedding with the possible translated word which could be our third input. Now we have a layer which have some percentage of ground truth about true translation.
- One English word could have multiple meaning in French, so overcome that randomness has been in choosing the translation in the dictionary.
- This new embedding layer has token with average sentence length as 22.90 whereas our English corpus had 25.73, which is very close. We can say that this new embedding represents the true translation of the French corpus.

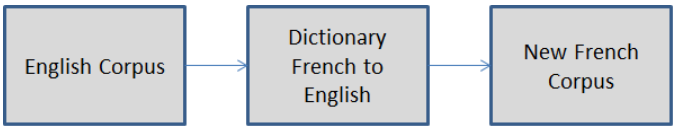

Fig. 2. New French Corpus

Improvement in layers:
- In the base line model, added six hidden layers with 64, 32, 16, 8, 4, 2 neurons respectively.
- The dropout of 50% after each hidden is also added as regularizer to control the training rate, and it did help in reducing the training accuracy from 89% to 74% and improves the model test accuracy 78.28%.
- There is no issue of over fitting here, since in the previous model from 64 neurons we were flattening them and passing the output of hidden layer to sigmoid function, but now the numbers of neurons are reduced slowly with the six hidden layers and then passed to the classification function.

TABLE III
CONFUSION MATRIX — IMPROVED BASELINE MODEL WITH ONE HIDDEN LAYERS

	Predicted lable '0'	Predicted label '1'
Actual '0'	65,431	34,569
Actual '1'	8,856	91,144

Results:
- The Recall is 91%, Precision 72% and F1 Score is 80%, it is very clear that the model improved way above our baseline. The number of miss classified data points for true translation from 43,977 down to 8,856.
- Recall expresses the ability to find all relevant instances in a data set, precision expresses the proportion of the data points our model says was relevant actually were relevant, reference [4]. So, from baseline was recall went up from 56% to 91%, which is significant and F1 score increased from 69% to 80%.

Interpretation:
- The problemwith this model is, the precision reduced from 83% down to 72%, is it acceptable ?, depends on the situation, in this model we are actually telling more lies than previous one for classification problem. Although the recall went up to 91% but there are more of wrong translation, which got labeled as true translation.
- But, when it comes to Machine Translation, this model could be better, because when user type English sentence, this improved model can give more accurate translation in French because of better Recall. So, there is a tread off between Precision and Recall depends on the problem we are solving.

*2) **Model 2 — Introduction of Bi-LSTM on Improved Baseline**:* There are couple of improvement we are expecting, first being better Recall and Precision, and also test whether add those two extra features is really required in Bi-LSTM of we can easily achieve improved baseline without them ?

*a) **LSTM Network**:* LSTM are explicitly designed to avoid the long-term dependency problem. The basic structure of LSTM memory unit is composed of three essential gates and a cell state.

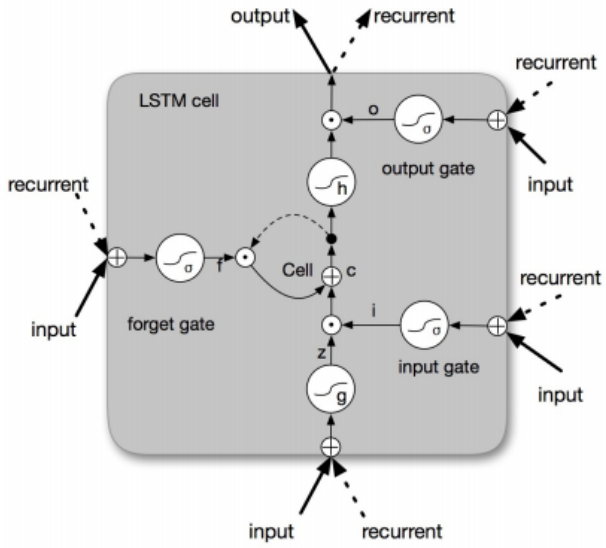

Fig. 3. Schematic of LSTM unit

Source of Figure 3 [5]

As shown in Figure 3, the memory cell contains the information it memorized at time t, the state of the memory cell is bound up together with three gates, the input vector of each gate is composed of input part and recurrent part. Forget gate controls what to abandon from the last moment, input gate decides what new information will be stored in the cell state, the output gate decides which part of the cell state will be output and the recurrent part will be updated by current cell state and fed into next iteration. [5]

The formal formulas for updating each gate and cell state are defined as follows:

$$z^t = g(W_z x^t + R_z y^{t-1} + b_z)$$
$$i^t = \sigma(W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b^t)$$
$$f_t = \sigma(W_f x^t + R_f y^{t-1} + p_f \odot c^{t-1} + b_f)$$
$$c^t = i^t \odot z^t + f^t \odot c^{t-1}$$

$o^t = \sigma(W_o x^t + R_o y^{t-1} + p_o \odot c^t + b_o)$
$y^t = o^t \odot h(c^t)$

Here $x^t \in R^d and\ y^t \in R^d$ and $y^t \in R^d$ are input and output vector of the unit at time t, $W_k(k = z,i,f,o)$ and $R_k(k = z,i,f,o)$ are weight matrices for input part and recurrent part of different gates, $b_k(k = z,i,f,o)$ denotes bias vector and the functions σ, g and h are non-linear functions such as sigmoid or tanh, $\odot$ mea denote peephole connection and is mostly used in LSTM variants. $ $p_k(k = i,o,f)$ [5]

*b) **Bi-LSTM Network**:* BLSTM network is designed to capture information of sequential dataset and maintain contextual features from past and future. Different from LSTM network, BLSTM network has two parallel layers propagating in two directions, the forward and backward pass of each layer are carried out
in similar way of regular neural networks, these two layers memorize the information of sentences from both directions. Since there are two LSTM layers in our network, the vector formula should be also adjusted.

$h_{f_t} = H(W_{xh_f} x_t + W_{h_f h_f} h_{f_{t-1}} + b_{hf})$
$h_{bt} = H(W_{xh_b} x_t + W_{h_b h_b} h_{b_t-1} + b_{h_b})$

$h_f \in R_d\ and\ h_b \in R_d$ denotes the output vector of forward layer and backward layer respectively, different from former research, the final output in our work $y_t = [h_{f_t}, h_{b_t}]$ is the concatenation of these two parts, which means $y_t \in R^{2d}$. We define the combination of forward and backward layers as a single BLSTM layer.

Why Bi-LSTM not LSTM or any other RNN?, the only reason is that, it is a state of art model to be applied in the field of machine translation for past couple of years. But the real reasoning is, for machine translation to work it need to see the current state and past steps, but also the future state which is equally important. The word's translation depend on its previous and future words, on daily basis, when we say any command (voice) to Siri (iPhone), it first listen the complete sentence we say, then it start reacting by printing that voice command into text.

Equation here:

Architecture :

- The feature space remain same with the similar parameters as of improved baseline model, we will keep our 6 hidden layers with 64, 32, 16, 8, 4, 2 neurons respectively, the reasoning is as last time.
- The Bi-LSTM is applied just after we merge our embedding, which returns output of two dimension which goes to the hidden layers and output of that passed to the sigmoid function.
- The only reason there is no change in any parameters is that it will help us to compare this model with our previous two models.

Results :

- The model accuracy on test is 83.38%, which is significantly higher than our baseline and 5% higher than our improved baseline model.
- The Recall here is 93%, Precision is 78% and the F Score is 85%. There is small improvement in Recall (1%) with significant improvement in Precision from our previous result.

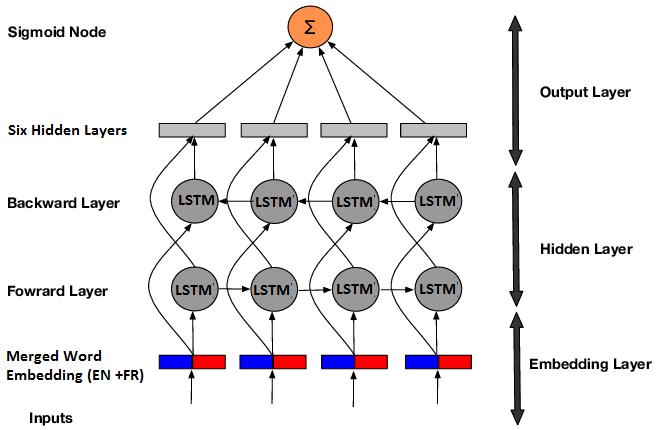

Fig. 4. Bi-LSTM Architecture

TABLE IV
CONFUSION MATRIX FOR BI-LSTM MODEL

	Predicted lable '0'	Predicted lable '1'
Actual '0'	73,799	26,201
Actual '1'	7,033	92,967

Interpretation:

- The model results are better than previous models but it did not perform as expected. Bi-LSTM is working extremely well in the ability to understand the true translation more than the negative samples. One reason could be the new French corpus is helping the model to understand the true vector space more and did not do well for negative sample.
- One more reasoning we can give here is, the negative sample is very similar to true sample, and it is becoming hard for the model to differentiate. Cosine distance (or any other distance measure) would help us to explain more about the vector space of negative sample compared to true translation. The solution for this could be, if we add more variant on negative sample in large number the model would perform better in deciding the boundary, here the data is balanced.

*3) **Model 3 — Introduction of CNN and LSTM**:* Based on our previous interpretation we should ideally look for more negative sample to train the model more efficiently, but trying an ambitious step here by applying CNN and LSTM together on the same feature to see if our interpretation about having more negative sample is correct.

Why CNN and LSTM together ?,

Convolutional Neural Networks (CNNs) are networks initially created for image-related tasks that can learn to capture specific features regardless of locality. But what about CNNs? CNNs are basically just several layers of convolutions with nonlinear activation functions like ReLU or tanh applied to the results.

So, how does any of this apply to NLP? Instead of image pixels, the input to most NLP tasks are sentences or documents represented as a matrix. Each row of the matrix corresponds to one token, typically a word i.e. each row is vector that represents a word. In our case, these vectors are the merged

word embedding 40 word sentences using a 300-dimensional embedding we would have a 40×300 matrix as our input.

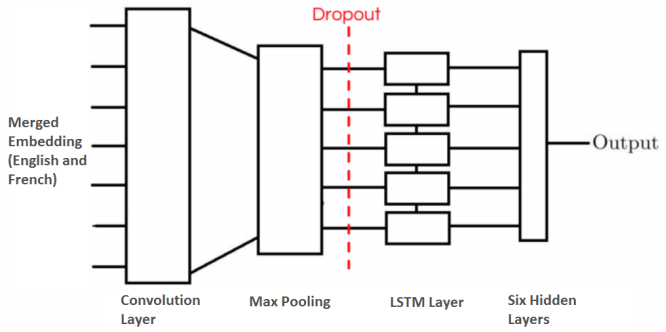

Fig. 5. CNN LSTM Architecture

Architecture:

- The features and all the hidden layers remains same for this analysis.
- CNN expect two very important parameters, batch size and channels, where batch size represent number of tokens and channel represents the embedding output. Filters parameters is just how many different windows you will have, all of them with the same length, which is channels.
- How many different results or channels we want to produce ?, we have used filters=64 and kernel size=3, we are creating 64 different filters, each of them with length 3. The result will bring 64 different convolutions.
- The output of above goes to pooling, key aspect of CNN are pooling layers, typically applied after the convolutional layers. Pooling layers sub-sample their input. The most common way to do pooling, it to apply a max operation to the result of each filter. Why pooling? There are a couple of reasons, one property of pooling is that it provides a fixed size output matrix, which typically is required for classification. Reference [6].

Results:

- The model accuracy on test data is 82%, which is very close to single layer Bi- LSTM model. Recall is 97%, Precision is 75% and F Score is 85%.
- The model improved significantly for true translation, we are correctly classifying 97,217 but there is a penalty on negative sample due to this.

TABLE V
CONFUSION MATRIX FOR CNN LSTM

	Predicted lable '0'	Predicted lable '1'
Actual '0'	66,908	33,092
Actual '1'	2,783	97,217

Interpretation :

- This model as we can see is very good at predicting the true translation but did not do well on negative sample. The same case was in LSTM as well, where the model is biased on true sample.
- We give the same reasoning here as well that, the negative sample is very close to the true translation and the model could improve if we add more variant to negative sample.

VI. CONCLUSION

The output of all model on the given test data in mentioned below, in all the model training and validation accuracy was close to the test accuracy. If we see the output, there is no model which is ideal for this scenario, although the accuracy is increasing if we have complicated model, but it comes with a cost of either precision or recall. The baseline model showed good classification on negative sample, but the rest three models are significantly good on recall with penalizing the negative sample accuracy.

TABLE VI
ANALYSIS OF ALL MODEL (*values are in %*)

Model	Accuracy	Precision	Recall
Baseline	71	83	56
Improved Baseline	78	72	91
Bi-LSTM	83	78	93
CNN-LSTM	82	75	97

Reasoning and Conclusion of model performance:

- The one important understanding we got by seeing all results are, the negative sample are very close to true translation, which usually the real life scenario. We can overcome this issues by providing mode negative sample to train the model
- The experiment was performed of FastText word embedding, and it might not represent the domain of the corpus. For machine translation, understanding domain is an important task, since weight (meaning) of each word changes with the context in which it's been spoken. It could be resolved by training the embedding from scratch with the corpus, this approach also has its own limitations such as size of data.
- The model performance could be further improved by improving the parameters and the feature space.

VII. FUTURE WORK

There is a saying in world of machine learning — "Simpler the Better", there are much simple approach with which this problem could be solved. One could be, just using the features like, sentence length and lexicon dictionary, it is easy to find distance between the parallel corpora which might give better accuracy for this classification problem, but the question is it a right way to solve a machine translation problem?

The Bi-LSTM or CNN-LSTM model parameter can be improved and efficiently adding regularizer could also lead to better accuracy.

REFERENCES

[1] F. Gregoire and P. Langlais, "Extracting Parallel Sentences with Bidirectional Recurrent Neural Networks to Improve Machine Translation." [Online]. Available: http://www.iro.umontreal.ca/~felipe/bib2webV0.81/cv/papers/paper-francis-coling-2018.pdf

[2] J. Brownlee, “How to Use Word Embedding Layers for Deep Learning with Keras.” [Online]. Available: https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/
[3] M. Schumacher, “Using FastText models for robust embeddings.” [Online]. Available: https://www.kaggle.com/mschumacher/using-fasttext-models-for-robust-embeddings
[4] “sklearn.metrics.precision_recall_fscore_support.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
[5] Y. Yao and Z. Huang, “Bi-directional LSTM Recurrent Neural Network for Chinese Word Segmentation.” [Online]. Available: https://arxiv.org/pdf/1602.04874.pdf
[6] “Understanding Convolutional Neural Networks for NLP.” [Online]. Available: http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/