Open Science Index, Computer and Information Engineering Vol:15, No:4, 2021 publications.waset.org/10011951.pdf

End-to-End Spanish-English Sequence Learning Translation Model

Vidhu Mitha Goutham, Ruma Mukherjee

Abstract**—**The low availability of well-trained, unlimited, dynamic-access models for specific languages makes it hard for corporate users to adopt quick translation techniques and incorporate them into product solutions. As translation tasks increasingly require a dynamic sequence learning curve; stable, cost-free opensource models are scarce. We survey and compare current translation techniques and propose a modified sequence to sequence model repurposed with attention techniques. Sequence learning using an encoder-decoder model is now paving the path for higher precision levels in translation. Using a Convolutional Neural Network (CNN) encoder and a Recurrent Neural Network (RNN) decoder background, we use Fairseq tools to produce an end-to-end bilingually trained Spanish-English machine translation model including source language detection. We acquire competitive results using a duo-lingo-corpus trained model to provide for prospective, ready-made plug-in use for compound sentences and document translations. Our model serves a decent system for large, organizational data translation needs. While acknowledging its shortcomings and future scope, it also identifies itself as a well-optimized deep neural network model and solution.

Keywords**—**Attention, encoder-decoder, Fairseq, Seq2Seq, Spanish, translation.

I. Introduction

FOR most of today's translation tasks, we tend to look to Google Translate. But when the requirement is much larger by scale - we will be faced with the absence of a plug-in option. Though translation seems like an easy task, there is a surprisingly huge void when it comes to the availability of functional and easy-to-use opensource translation utilities. The only reliable one available will not be an option unless you are willing to shell out a lot of money. Google is the standalone player, sole provider, and beneficiary at the moment in this area.

II. Availability

The higher the organization's need for precision, the steeper the price. All the available (precision) techniques charge on a per-character basis. Google even charges per-access of the *detect* method call, i.e. for identification of each word's source language and then separately for each character's translation.

The only other alternative would be to opt to self-train a model with a dataset to perform the task. For tailor-fit translation needs, we would then look to train available models on domain-specific data of the source language and target language. But all the current market models are either amateur or still underdeveloped and filled with bugs and outliers. Additionally, these tend to be highly conditional and constrained utilities. Other generic approaches such as using Application Programming Interfaces (APIs) cannot be employed for larger sets of data requiring translation, as all of these APIs are privately monitored and a cap is set on the number of hits or accesses per IP address call. Beyond a small number of free accesses, that too turns chargeable. So for unlimited access, there is currently no open-source availability.

In most global organizations with international clients, there are multiple avenues and products requiring translation. And each product's components would be built to hold reusable prototypes. In such scenarios, there is an absolute requirement for translation operations to be accessed innumerable times, especially concerning the model's training. Present utilities are only suitable for niche requirements and the associated costs make it infeasible in terms of functionality. Hence there arises a need for an end to end solution that is openly deployable.

III. Environments and Plausibility

Deployment concerns usually start and end with security. Most translation solutions are cloud-based due to the need for embedded training and testing utilities. But as convenient as cloud platforms could be, it comes with a huge range of vulnerabilities and overheads. Without a compromise on that factor, incorporating these plugin models is a major concern.

Organizations usually value and are bound by high privacy requirements. These translation tools used in products need to be exclusive to the client while being internally reusable and would require utmost confidentiality, copyrights, etc. which makes it incompatible with most current environments.

Facilitating transfer learning is to be factored in as well to ensure that model functionality can be used across clients within the organization. It would, therefore, require a private cloud or service to be developed or rented specifically for these needs. Infrastructure expenditure attributes to the highest weigh-down factor in any company's spending or budget and virtual machines in data centers and cloud storage incorporate heavy infrastructure costs. Hence it increases the overall infeasibility.

There is also an inherent lack of datasets for many non-English languages. And domain-specific ones are even scarcer to find. The option of generating such datasets has its shortcomings. They tend to be hard to produce and grammatically incorrect which would then mistrain our model and accuracy will be stunted very low.

Another major incompatibility issue arises due to the format

Vidhu Mitha Goutham is with the Unisys, India (e-mail: vidhu.mitha@gmail.com).

and nature of the text. In some cases, it comprises of millions or billions of rows of data in various formats. This would require a lot of intermediate coding for access and conversions, which is a time and resource-consuming task.

IV. Literature Survey: Translation Modules

A. Usual Shortcut Solutions- APIs

1) *Googletrans and Natural Language Tool Kit (NLTK):* It is an unofficial library using the web API of translate.google.com with a maximum character limit on texts at 15k**.** This API does not guarantee, due to limitations of the web version of Google translate, that the library would provide correct output translation or predict concisely in all scenarios. This may be more suitable only for domestic day-to-day translation- leaving it inadequate for organizational translation needs.
2) *Goslate:* It acted as a free python API querying the Google translate website. Google translation does not support very long text, and Goslate simply bypasses this limitation by splitting the long text internally before querying Google and then joining the results back into one translation text. Its functionality was hinged on an intricate loophole and hence was not a stable method. Additionally, Google updated its translation service in the recent past with a ticketing mechanism to prevent crawler programs - rendering Goslate inept for translation in comparison.
3) *Google's Translation API:* It helps make a cloud translation using a REST method call to the basic translate method. But this only applies to usage on terms or phrases. It requires a lot of self-customization before passing larger input data, or for domain and context-specific translation. This too is accompanied by a price tag.

B. Detection Modules

1) *Langdetect:* This is a direct port of Google's language-detection library from Java to Python. It serves a very limited number of languages and simply performs detection of the source language. The language detection algorithm is non-deterministic, i.e. if you try to run it on a text which is either too short or too ambiguous, you might get different results each time you run it [2].
2) *Facebook's-fastText:* It is an efficient pre-trained model that performs accurate identification of source language. It supports a whopping 170+ languages but is licensed for use. Additionally, the model was trained on UTF-8 data, and therefore strictly needs UTF-8 inputs.

C. Translation Using a Model

1) *Neural Machine Translation- Sutskever Model:* It is a proposed Encoder-Decoder model for machine translation. It uses sequence-to-sequence learning on a multilayered Long Short-Term Memory (LSTM). This theory provided a base concept for building a model that, for the very first time, could map the input sequence to a vector of fixed dimensionality, and then decode the target sequence from the vector. It performed better than a phrase-based SMT (Statistical machine translation) system on the same datasets. The upcoming-Google's successful translation system is also based on this very modeling prototype [3]-[6].
2) *Google's AutoML Translation:* A service available only upon payment, it creates production-ready models. But it requires us to upload cleaned, pre-processed, and translated language pairs, a task that is quite tedious.

V. A Smarter Approach

We finally found that constructing a model and coupling it with a handpicked dataset for training yields some impressive results. So we built a system that performs end-to-end: identification and translation. We used a combination of Fairseq and the Sutskever technique. Fairseq is a toolkit written in PyTorch for sequence modeling that allows developers and researchers to custom-build and train models for summarization, language modeling, and text generation.

A. The Model

Our base model is a Deep Neural Network Sequence-to-Sequence model. It maps a fixed-length input with an output where the length of the input and output may vary or differ dynamically. We used an LSTM architecture as LSTM models are fairly easier to train. The model comprises of 3 major parts: encoder, intermediate vector, and decoder:

- Encoder: holds a stack of recurrent units of LSTM or GRU cells where they accept each individual element of the input sequence, collect information for that element, and then moves it forward.
- Decoder: predicts an output *y_t* at a time step *t*. Each recurring unit accepts a (hidden) state from the previous one. The output sequence is then collected as a set of all words from that specific answer. Each word in this case is represented as *y_i* where *i*- represents the order of that word.
- Formula to compute the hidden states:

$$h_t = f\,(W^{(hh)}h_{t-1} + W^{(hx)}x_t) \qquad (1)$$

Since ours is a Seq2Seq model consisting of a CNN encoder and an RNN decoder, the encoder will extract features from the written text line and sequentially encouple temporal context between the input sequence and the integer dictionary. While using the attention mechanism to focus on the most relevant encoded features, the decoder outputs a sequence of characters. It can thus be used end-to-end on complex large lines as well as files.

We have also used SentencePiece- as an unsupervised text tokenizer and detokenizer. Apart from performing direct training from raw sentences, it also allows us to create an end-to-end system that does not depend on language-specific preprocessing or postprocessing. It backs two segmentation algorithms: byte-pair-encoding (BPE) and unigram language model. It basically treats the input text as a simple sequence of Unicode characters which would make it easier to train as data input sentences can be pre-tokenized or raw sentences.

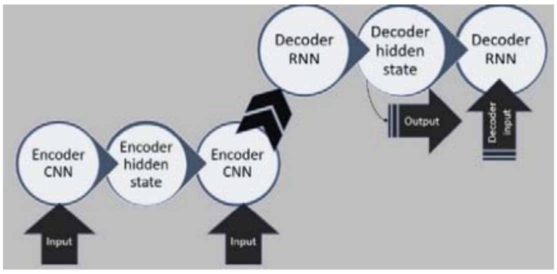


Fig. 1 Hidden state flow impacting translation

B. The Dataset

We used the Tatoeba corpus of bilingual sentence pairs. We trained our model on a set of 1.56 Lakh sentences which includes 1.44 Million English words and 1.56 Million Spanish words. This dataset has also been cleaned to have tab separation spaces and full stops at the end of every sentence. Additionally, as typical neural language models rely on a vector representation for each word, we used a mapped dictionary for both languages.

C. Pre-Processing and Training

We used Google Collab as the training environment due to the availability of a combination of GPUs and processing memory. Fairseq contains utilities such as an output generator-accessible through a command-line interface. So at the preprocessing step, we call the wrapper, which internally calls the python preprocess.py method. In this method of the defined class for data, we unzip the dataset to Collab, tokenize each sentence, and proceed to convert word sentences and map it to their corresponding integer dictionary. We used tokenizer Space in order to tokenize each string by tabspace. We then assign designated test, train and validation split sets. These were derived for Spanish and the mapped English data while maintaining consecutive row-wise translation mappings. Then the flow proceeds to write the binarized (& tokenized) data [7]-[10].

We added a source threshold of 3 to ensure efficient mapping of words appearing more than three times in the whole training set- to at least a synonym prediction and words appearing less than 3 times to an unknown tag to demarcate unavailability of the unique word's translation.

We used the Adam optimization algorithm since it facilitates the network weights to be updated iteratively during training. Our LSTM architecture also contributed in this aspect due to its characteristic feedback connections as it can process entire sequences of data while maintaining dynamic development. This ensures that the dynamic increase and progress of model weights are constantly recorded and saved. It allowed us the liberty of further picking the best checkpoint weights that could lead to better translation. Our optimizer also adjusts the model with respect to loss function to get the most accurate weights. We set it to share embedding weights within encoder and decoder as its ongoing research that proves output embedding and sharing – in fact improves language or text related NLP models.

We trained the model for around 1066 epochs and saved every 5th checkpoint i.e. at every 5th epoch. The weights at each epoch are further compared to the current standing best_checkpoint weights file and replaced with the current checkpoint's weights if it is better.

We fixed a learning rate of 0.001(i.e. 1.0E-3). We noticed that the loss dropped dramatically past the 230th epochs and was hitting lower than 0.138 and ~0, with more epochs of training. We used batches set to our max token size: 4096.

We used label_smoothed_cross_entropy as our criterion to compute loss function constantly while avoiding overfitting of the model. We set our beam value to 5 to ensure the translation could be varied by length wherever necessary. At every step, the network produces a probability distribution over the next possible tokens.

We used the Attention Mechanism to retain focus only on the most relevant features encoded at each decoder's step. Our attention layer creates a dynamic mapping between each of the hidden states (at the encoder's end) and decoder output by having access to the whole input sequence and picking out elements of higher relevance. Without the attention mechanism, the last hidden state value of the encoder is internally passed as the context vector for the decoder.

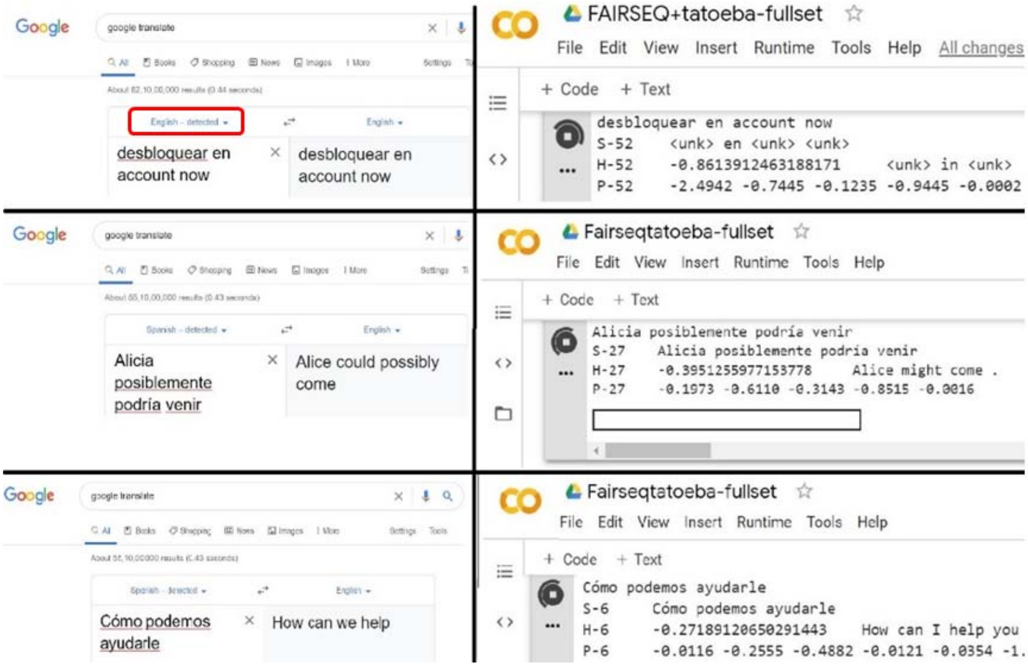
VI. Result

To perform end-to-end we added a component to input the data and predict its language using the IsEnglish functionality. Upon identifying the language, and in our case the model is heavily trained to perform Spanish translation, the control is then passed to the translation utility [1]. As monolingual models have a homogeneous functionality we utilized Fairseq to train a Bilingual model, so it will translate input Spanish strings to English as well as predict the Spanish string for English inputs [11], [12].

Our outputs, though effective, will be a foreigner to new words and slang. Therefore, words not present in our dataset will currently display the special <unk> token. This is where training on domain-specific datasets makes a substantial difference in the output translation's precision.

1) *Output 1 shows us*- Google Translate's misdetection of a mixed Spanish sentence as English itself and the translation as the string remaining as it is, untranslated. For the same string, our system detected it as a Spanish string but displays the translation of certain words as <unknown> while translating the word in the middle to English.
2) *Output 2 shows us*- an interpretation of the string done correctly by both systems but varying in terms of redundancy as in Google Translate's output while our system avoided that problem.
3) *Output 3 shows u*s- a correct translation by both systems - differing only by a small degree in terms of casual language interpretation.

And though Google Translate is still learning new words as well, it remains as a superior competitor. In comparison, our model performs satisfactorily and is a cost-free opensource solution.



desbloquear en account now

desbloquear en account now

```
desbloquear en account now
S-52    <unk> en <unk> <unk>
H-52    -0.8613912463188171     <unk> in <unk>
P-52    -2.4942 -0.7445 -0.1235 -0.9445 -0.0002
```

Alicia posiblemente podría venir

Alice could possibly come

```
Alicia posiblemente podría venir
S-27    Alicia posiblemente podria venir
H-27    -0.3951255977153778     Alice might come .
P-27    -0.1973 -0.6110 -0.3143 -0.8515 -0.0016
```

Cómo podemos ayudarle

How can we help

```
Cómo podemos ayudarle
S-6     Cómo podemos ayudarle
H-6     -0.27189120650291443    How can I help you
P-6     -0.0116 -0.2555 -0.4882 -0.0121 -0.0354 -1.
```

Fig. 2 Result comparison- Google Translate (column 1) & our system (column 2)

VII. Future Scope

Statistically, all current opensource translation methods continue to contain shortcomings. Glitches become unavoidable because contextual inconsistencies are tricky in modern language and scripts.

- By tuning the attention mechanism further with twitter-like language, it will help identify sarcasm, slang as well as the deterministic word for such cases and place them in the right context.
- Mapping such keywords will increase and magnify grammar detail as well.
- An additional enhancement is coupling our model with audio inputs to perform speech-to-text translation. Due to high optimization, our model would be a good fit.
- Using a much larger or varied dataset, inclusive of social media data or narrowed domain-specific data, would help improve our decoder's output.
- Using a dataset of much higher size and training it for about 1000 more epochs would also benefit as the model has the capability to improve until its capping precision.
- An alternate approach that could produce an interesting change is the reversal of the order of the input sequence (improvement at encoder). Further, adding varied sets of GRU and LSTM placement in the architecture may facilitate improvement.

References

[1] T. Strauß, "*Decoding the output of neural networks - a discriminative approach*," Ph.D. dissertation, University of Rostock, 2017.

[2] Ron J. Weiss, Jan Chorowski, Navdeep Jaitly, Yonghui Wu, Zhifeng Chen "*Sequence-to-Sequence Models Can Directly Translate Foreign Speech*" arXiv:1703.08581v2 (cs.CL) 12 Jun 2017

[3] J. Poulos and R. Valle, "Attention networks for image-to-text," *CoRR*, vol. abs/1712.04046, 2017

[4] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *Proceedings of ICASSP*, 2017.

[5] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 12 2014.

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in Neural Information Processing Systems*, 2014, pp. 3104–3112

[7] A. Sriram, H. Jun, S. Satheesh, and A. Coates, "Cold fusion: Training seq2seq models together with language models," *CoRR,* vol. abs/1708.06426, 2017

[8] Ofir Press and Lior Wolf "*Using the Output Embedding to Improve Language Models*" arXiv:1608.05859v3 (cs.CL) 21 Feb 2017

[9] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, 05 2010, pp. 249–256

[10] G. Kumar, G. W. Blackwood, J. Trmal, D. Povey, and S. Khudanpur, "A coarse-grained model for optimal coupling of ASR and SMT systems for speech translation." in *Proceedings of EMNLP*, 2015, pp. 1902–1907.

[11] E. Vidal, "Finite-state speech-to-speech translation," in *Proceedings of ICASSP*, vol. 1. IEEE, 1997.

[12] F. Casacuberta, H. Ney, F. J. Och, E. Vidal, J. M. Vilar, S. Barrachina, I. Garcıa-Varea, D. Llorens, C. Martınez, S. Molau et al., "Some approaches to statistical and finite-state speech-to-speech translation," in *Computer Speech & Language*, vol. 18, no. 1, pp. 25–47, 2004.