Bachatero's thesis

Physics Engineering, ETSTB

Universitat Politècnica de Catalunya

Combining Multilingual Machine Translation and other NLP tasks to Learn Intermediate Language Representations

(Enhancing Intermediate Representations by Jointly Learning Multilingual Translation and Part-Of-Speech Tagging)

Author: Júlia Sánchez Martínez

Advisors: Carlos Escolano and Marta R. Costa-jussà

June 28th 2021

Abstract

In a world in which the internet gives us access to any kind of information, there are still limitations when the source of such information is presented in another language. Online translators are far from perfect, this is why language machine translation is a trending topic in the field of deep learning.

The purpose of this project is to use the Transformer architecture, developed by Google in 2017, in the context of Multilingual Machine Translation and to improve its results both in translation and a common intermediate representation.

The Transformer model is focused on self-attention and composed by an encoder and decoder that rely on a common intermediate representation of the source language. For the purpose of raising the BLEU score that defines the quality of the translation and enhancing the common intermediate representation, we have introduced Part-Of-Speech tagging in the encoder of the model. We perform experiments with four languages (English, Spanish, French and German) both in Machine Translation and in Cross-lingual Natural Language Inference. Finally, we visualize the intermediate representation and make experiments to see how source embeddings codify gender information.

Comparing a baseline model without tagging with the new POS tagged codes, the translation BLEU has decreased 0.50 points on average. In the case of NLI, the accuracies have also decreased 8% on average, showing that the POS tagged models do not improve the performances of these tasks. However, in the gender experiments of the encoder embeddings, the accuracy of the gender classification for professions has increased by 1.1%.

Keywords: Machine Learning (ML), Machine Translation (MT), Natural Language Inference (NLI), Part-Of-Speech (POS) Tag, Neural Network (NN), Baseline Model, Attention Mechanism, Natural Language Processing (NLP), Statistical Machine Translation (SMT), epoch, loss function, Byte Pair Encoding (BPE).

Acknowledgements

First and foremost, many thanks to my project tutors Carlos Escolano and Marta R. Costa-Jussà, who have helped me and encouraged me during the whole bachelor's degree thesis. Also, thank you very much to Christine Basta, who was always present in the group meetings and has helped me understand many concepts regarding the Transformer code.

Finally, I would like to thank all the members of the Machine Translation[1] research group of the UPC (Universitat Politècnica de Catalunya) who are leading an extraordinary investigation, which I am sure will achieve amazing things.

[1] https://mt.cs.upc.edu/people/

Table of contents

List of figures

List of tables

CHAPTER I

1 Introduction

The use of neural models in the field of Machine Language Translation, which consist on using neural networks to build end-to-end translation systems, has increased dramatically in recent years (Peris, Domingo, & Casacuberta, 2016) as it has achieved state-of-the-art performances in large-scale translation tasks (Luong, Pham, & D.Manning, 2015). Most models are based on an encoder-decoder architecture that jointly trains language pair datasets (source and target languages) to maximize the probability of getting the correct output. First, the encoder reads the source data, which is mapped into an intermediate space representation that is then decoded to generate a correct translation into the desired target language (Bahdanau, Cho, & Bengio, 2016) .

Universal encoders and decoders are not the only approach to multilingual translation tasks, as language-specific ones offer some advantages over them. Shared encoders/decoders for many-to-many translation mainly have three problems. Firstly, when a new language is added, the whole system has to be retrained, which can take a lot of time. Also, the translation quality drops for the languages that have the most resources or when the system contains too many language pairs (Arivazhagan, et al., 2019). Finally, regarding the previous issue, when many languages are inserted, especially in the case they have different alphabets, the shared vocabulary grows a lot. With the language-specific approach that we will use during the thesis, the aforementioned limitations will be overcome. We will train separate encoders and decoders for each of the available languages without sharing any parameters across modules. This allows to introduce new languages without having to retrain the entire system. (Escolano, Costa-jussà, Fonollosa, & Artetxe, 2020).

The Transformer network architecture we have used in the project is based on the Fairseq implementation[2]. This translation model that relies solely on attention mechanisms has proven to outperform recurrent and convolutional neural networks since it is more parallelizable and requires less time to train (Vaswani, et al., 2017).

2 Goals

The main goal of the thesis is to enhance the source language encoded representation by adding Part-Of-Speech (POS) Tags in the encoder of the Transformer model. We have implemented an additional layer that predicts Part-Of-Speech Tags of the input sequence and

[2] https://github.com/pytorch/fairseq

that will be trained jointly with the translation task. Natural Language Inference (MacCartney, 2009) is the task that we have employed to test the quality of the intermediate representation as well as computing the translation quality by means of BLEU (Papineni, Roukos, Ward, & Zhu, 2002) for each language pair. As a qualitative measure, we have also used the visualization of the intermediate representation. Finally, with the aim of measuring the effect of the POS Tags implementation in the encoder's embeddings, we have made experiments relating contextual embeddings and gender information.

3 Contributions of the thesis

The present thesis introduces Part-Of-Speech tagging to a multilingual translation system in order to enhance the common intermediate representation in which the system relies on as well as improving the translator performance in different tasks. This architecture achieves to better determine gender from the encoder embeddings in the case of determiners and professions.

4 Thesis organization

The thesis is organized in 5 chapters.

- **Chapter 1:** Introduction to the project, goals, work plan and used tools.
- **Chapter 2**: Background on Machine Learning, Neural Machine Translation and concepts that appear during the thesis. Additionally, it contains a theoretical framework of the architecture and composition of the baseline system.
- **Chapter 3**: Code implementation, data, training and setbacks during the training.
- **Chapter 4**: Experimental framework. Experiments performed with the developed models and results of their performance.
- **Chapter 5**: Conclusion and discussion of the results.

5 Work plan

In this section we explain the different parts in which the thesis was developed and structured.

- **Research and installation:** During this first phase, we gathered information about the Transformer architecture and we reviewed the state of the art in shared and language–specific encoders and decoders too. Furthermore, the configuration of the computer was carried out by installing all the necessary programs, toolkits, repositories and libraries.
- **POS Tag implementation:** This is the part of the thesis that required most of the time, since we had to thoroughly investigate the provided baseline code, which was very large, and find libraries that could perform Part-Of-Speech tagging. Finally, we applied the POS Tags in the encoder and we downloaded the necessary libraries containing this feature.

- **Model Training and experiments:** Training all the different models was quite long, so as they were training, we did the experiments. We computed the translation BLEU score, we made Natural Language Inference experiments and the visualization of the intermediate space representation. In addition, we evaluated the embeddings of the encoder in gender information.

6 Tools

In order to initiate the project different development and monitoring tools were necessary, which we account for in the following.

6.1 Development tools

Python 3, specifically the 3.6 version, has been the programming language we have employed and Anaconda[3] was used as package and environmental manager. We developed the code locally with the PyCharm[4] program, which contains many features dedicated to ease Python programming and help debug scripts.

As previously mentioned, we used the Fairseq implementation of the Transformer model in the training and experiments, which is written in Pytorch. This is an open-source and python-based machine learning framework, developed by Facebook's AI Research, that is currently used in many computer vision and language processing applications. Pytorch, based on the Torch library, has been designed with the purpose of replacing NumPy library to use GPU power and other accelerators and also facilitate the implementation of neural networks. (PyTorch, 2021) (JournalDev, 2010)

6.2 Monitoring tools

We have used the Git control system, which allows you to have multiple independent local branches, to handle the project and keep track of the changes induced to the code. The Git repositories are stored in GitHub so that the code variations can be downloaded from different directories. This was useful since we did the code changes locally and then we send them to a server for training and testing.

[3] https://docs.anaconda.com/anaconda/
[4] https://www.jetbrains.com/es-es/pycharm/

CHAPTER II

7 Background

In this chapter we present an introduction to Machine Learning and Neural Machine Translation. Also, the intermediate space representation which will have a main role in this thesis is discussed in the concepts section together with other notions.

7.1 Machine Learning and Neural Machine Translation

Machine learning plays a main role in understanding a set of data and fitting it into a model so that it can be used by people. The algorithm, instead of being a set of explicitly programmed instructions, allows computers to train on data (training set) and build models to automate processes based on input data (Tagliaferri, 2017). The data set used to test the model and see how well it performs is called the test set.

Natural Language Processing is one of the many applications of the sub-branch of artificial intelligence that is ML. It enables machines to understand and extract meaning from natural human languages and helps in a lot of tasks, such as Machine Translation (Yse, 2019).

Neural Machine Translation (NMT) is state of the art in MT, developed to translate text from a source to a target language. Unlike traditional Statistical Machine Translation, NMT attempts to build and train a single neural network that reads input sentences and computes the correct translation inspired by the neural interconnection of a human brain (Bahdanau, Cho, & Bengio, 2016). Neural models are composed of an input layer, M hidden layers and an output layer of neurons. Each connection between neurons is associated to a weight, and the output of a hidden layer is computed as (Wang, 2003):

$$h_i = \sigma\left(\sum_{j=1}^{N} V_{ij} x_j + T_i^{hid}\right) \qquad (1)$$

The activation function, introducing non-linearity and bonding the value of neurons to avoid divergence, is defined as σ and the number of input neurons as N. V_{ij} are the weights, x_j the inputs and T_i^{hid} the threshold terms of the hidden neurons (Wang, 2003).

Concretely, a feedforward neural network is the neural model used in the Transformer architecture. In this network a function (f) is wanted to be approximated and information flows forward through the function being evaluated from the input x, through the intermediate computations defining f and the final output y, never forming a cycle. The output of the model never is fed back to itself (Gupta, towards data science, 2017).

For training feedforward networks, backpropagation is a commonly used algorithm, which is implemented in our Transformer model, for efficiently computing gradients and updating weights so that the loss function of the system can be minimized. This algorithm computes derivatives just using one forward pass through the network and one backward pass. The backward pass recursively applies the chain rule, staring at the end, to compute gradients (Gupta, towards data science, 2017).

7.2 Concepts

- **Common intermediate representation:** The Neural Machine Translation system we have employed to generate translations in this work is based in a common intermediate space. In this space, sentence meanings are aimed to be represented independently of their origin language so that similar phrases in different languages are represented closely. The intermediate space represents sentences from the output of the network encoders so that the original sentence can be recovered as well as the translation to another language with the decoder (Escolano, Interlingua based Neural Machine Translation, 2018).
- **Byte Pair Encoding (BPE)**: It is a word splitting technique used in Machine Language Translation to approach the open-vocabulary problem in translation by encoding unknown words as subwords tokens of the original word. Originally, BPE was developed as a data compression technique in which the most repeated pairs of bytes are replaced with single unused bytes. In the case of Machine Translation, words are divided into subwords chained to a special end-of-word symbol ('@@') instead of merging frequent byte pairs. The end-of-word character allows the retrieval of the original tokens after translation. An example of Byte Pair Encoding would be the splitting of the word 'lower' into 'low@@ er'. (Sennrich, Haddow, & Birch, 2016)
- **Part-Of-Speech Tagging:** In Natural Language Processing, POS tagging is the task of assigning Part-Of- Speech labels, with grammatically similar properties, to each token of a text. Generally, for each language the POS Tag set is different, but some examples in English are verb, noun, adjective, preposition... POS Tags are useful when recognizing lexical patterns and also to distinguish the case when a same word can be assigned a different category depending on the context. This happens with the word *work* in English, since it can be used as a noun or a verb (SketchEngine, 2018).
- **Accuracy:** The accuracy is a commonly used metric in classification problems that is useful when wanting to see how many cases the model correctly predicts. It is defined as the number of correct predictions divided by the total predictions the system performs. (Aprendizaje Automático, 2020)

$$Accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ total\ predictions} \qquad (2)$$

❖ **BLEU:** The BLEU, or Bilingual Evaluation Understudy, is an automatic evaluation translation method used when frequent and fast evaluations are required. It is a 0 to 1 range score that compares a reference sentence with a generated translation, token by token, being 1 a perfect match and 0 a total mismatch. The perfect match is rarely achieved, since it would mean that reference and generated translation are identical and not even a human translation would achieve such a score. (Papineni, Roukos, Ward, & Zhu, 2002)

The BLEU first computes the n-gram (sequence of n words) modified precision score accordingly to:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{Candidates\}} \sum_{n-gram' \in C} Count(n-gram')} \qquad (3)$$

A brevity penalty (BP) factor is also introduced into the definition, as in the precision only the first n-gram sentence by sentence matches are considered and it could happen that the first n-grams are correctly translated but not the rest. Taking *c* as the length of the candidate translation and *r* the effective length of the reference corpus, BP is computed as:

$$BP = \begin{cases} 1, & if\ c > r \\ e^{(1-r/c)}, & if\ c \leq r \end{cases} \qquad (4)$$

The BLEU's final formula, being w_n positive weights, is:

$$BLEU = BP \cdot exp(\sum_{n=1}^{N} w_n \log\ p_n) \qquad (5)$$

8 Theoretical framework

In this part, we introduce the concept of attention as well as the Transformer model which is solely based on it. Moreover, we will explain the Fairseq implementation of the used architecture in the thesis.

8.1 Attention and Transformer model

Ever since the concept of "Attention" was brought up in the papers (Bahdanau, Cho, & Bengio, 2016) and (Luong, Pham, & D.Manning, 2015), the quality of MT systems has highly improved. This mechanism consists of selectively paying special attention to the relevant parts of the input text during translation amplifying its signal (Alammar, Visualizing A Neural Machine Translation Model, 2018). As each position in the input sentence is processed, other positions in the input are looked for clues that can lead to a better encoding of the word (Alammar, The

Illustrated Transformer, 2018). Unlike classic sequence-to-sequence models, attention-based ones pass all encoder hidden state information to the decoder.

Attention is All You Need (Vaswani, et al., 2017) is the paper in which the Transformer model was described for the first time. It is a neural network architecture based on multi-head attention, composed by an encoder and a decoder stack and is especially well suited for language understanding (Uszkoreit, 2017).

Broadly speaking, it can be considered that an input sequence **x**= (x1, ..., xn) is fed to the encoder, which is then mapped into a continuous representation **z**, which is then given to the decoder to generate an output **y**= (y1, ..., ym) of symbols one element at a time. The whole model is auto-regressive and consumes previous decoder outputs in future steps (Vaswani, et al., 2017).

At first, the source text that is wanted to translate is turned into embedding vectors of dimension d. In order not to lose the words order of the text sentences, "positional encoding" is also added to the input embeddings following a specific pattern. This step is only applied before the bottom most encoder and decoder layers. Following the embedding, the source tokens enter the encoder similarly as they do in the decoder (Alammar, The Illustrated Transformer, 2018).

Next, we will explain the two main parts of the Transformer in detail.

- **ENCODER:**

A stack of N identical encoder layers composes the whole encoding block, all identical in structure but with different weights, the previous one feeding the next one. Each of these encoder layers also break down into two sub-layers, first a self-attention one and then a feed-forward neural network. After each of the previous layers, a layer normalization follows, meaning that the output of both encoder's sub-layers is $LayerNorm(x + Sublayer(x))$ (6).

The attention function takes as inputs a series of queries, keys and values sets and computes the following output matrix:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \qquad (7)$$

The $\sqrt{d_k}$ is a scaling factor, being d_k the dimension of the queries and keys.

Instead of computing the attention function only once, multi-head attention, consisting of several subblocks called heads, computes it h times with different weight matrices. The transformer computes the heads and then concatenates and projects the weights giving the final values.

$$MultiHead(Q,K,V) = Concat(head_1, \dots, head_h)W^O$$

$$\text{With } head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right) \quad (8)$$

And the parameter matrices as

$$W_i^Q \in R^{d_{model} \times d_k}, W_i^V \in R^{d_{model} \times d_v}, W_i^K \in R^{d_{model} \times d_k}, W_i^O \in R^{hd_v \times d_{model}}$$

The resulting matrix is then passed along to a fully connected feed-forward neural network and the same NN is applied to each position independently. (Alammar, The Illustrated Transformer, 2018) (Vaswani, et al., 2017)

- **DECODER:**

There are also N decoder layers, but in this case, they are composed of three sub-layers. Apart from the two described in the encoder, the decoder contains an additional layer performing multi-head attention over the final output of the encoder. Actually, the output of the encoder is transformed into attention vectors K and V that are fed to this layer, which helps the decoder focus on important parts of the input text. The Q matrix is created from the layer below it.

Before the first decoder layer, embedding and positional encoding is also added like in the encoder.

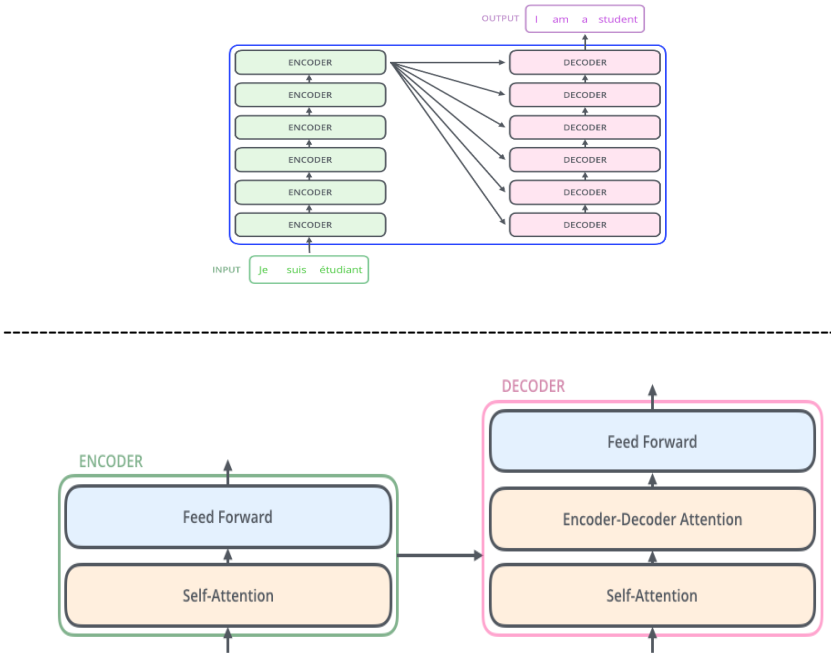

Figure 1: *Sketch of the encoder and decoder stacks (above) as well as the sub-layers they both contain (under) (Alammar, The Illustrated Transformer, 2018)*

In the final step of the decoder, a vector of floats is turned into representations of tokens by applying a Linear layer and a Softmax Layer to it. The linear transformation consists of a Neural

Network that projects the decoder vector into a larger vector of scores. The Softmax function then turns the scores into probabilities and what is normally done is taking the highest, which is associated with a token, as the output in that time step. (Alammar, The Illustrated Transformer, 2018) (Vaswani, et al., 2017)

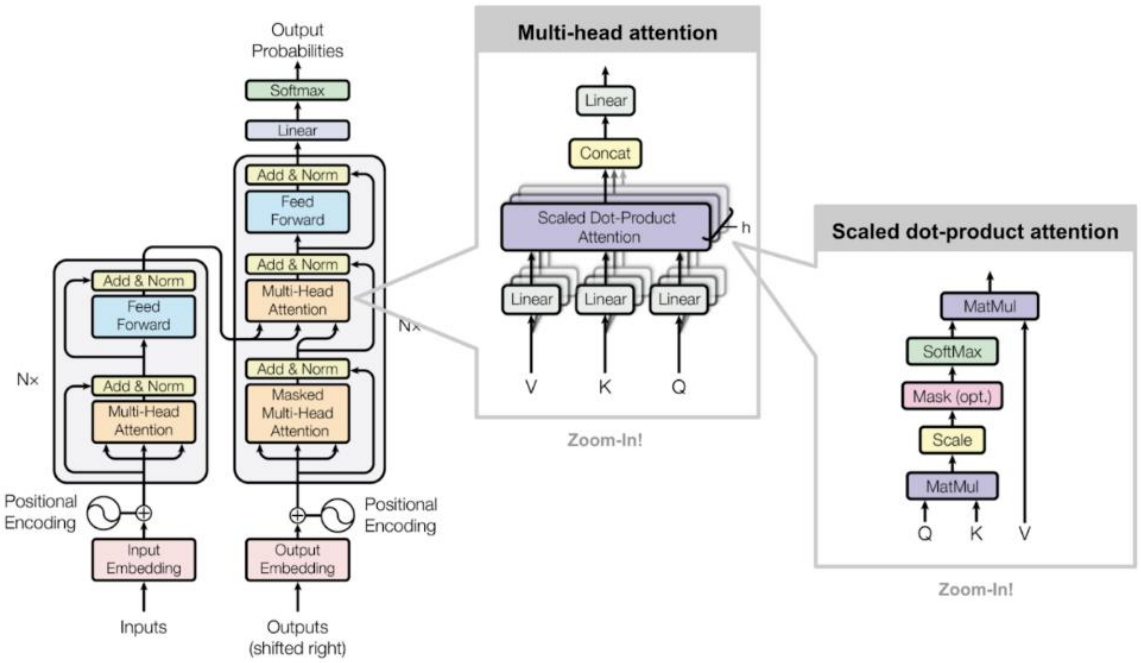

***Figure 2**: Transformer diagram (Baptiste Amato, 2019)*

8.2 Fairseq and baseline architecture

Fairseq is a PyTorch written toolkit, developed by Facebook AI Research, destined to help developers train a custom model for text generation tasks such as translation. Five different kinds of plug-ins can be added to Fariseq in order to extend models (Facebook AI, 2018). In the following, we summarize the plug-ins as well as the implementation for the baseline Transformer we have used:

- **Models:** they define the NN architecture and encapsulate all of its learnable parameters. In this project we have implemented the Transformer model, which includes the encoder and decoder stacks explained before.
- **Criterions:** These compute the model's loss function when fed the Transformer's output and the labelled target data. The loss function is the metric optimized during the training to estimate the parameters **Θ** of the model and achieve an accurate translator (Alammar, The Illustrated Transformer, 2018). We have used a label smoothed Cross-entropy as loss function in the baseline code.

Label smoothing has been used in language translation as a way to avoid the model becoming over-confident. It helps, for example, in case any target label data is wrong. The cross entropy is computed with soft targets, rather than the hard ones, which are a weighted mixture of these targets. (Müller, Kornblith, & Hinton, 2020)

- **Tasks:** The tasks are very important since they help initialize the Model, the Criterion, load Datasets and also store dictionaries. In this part of the code, we have loaded the Europarl dataset used in the training of the system.
- **Optimizers:** When trying to minimize the loss function and computing the gradients, we have used the Adam optimization algorithm to update the parameters of the model in order to find the best set. Adam is a method for efficient stochastic optimization and requires only first-order gradients. (Kingma & Ba, 2017)
- **Learning Rate Schedulers:** During the training, they update the learning rate. In our model, for the first warm up training steps, the learning rate is linearly increased and after it is decreased proportionally to $update\_num^{-0.5}$.

9 Methodology

Factored neural machine translation architectures take into account factored representation of words regarding linguistic aspects, such as Part-Of-Speech tagging, and also produce them as outputs. This neural approach enlarges remarkably the vocabulary and in turn decreases the number of unknown words. (Mercedes Garcıa-Martınez, 2016)

The aim of the project is to make a more coherent common intermediate space in which the encoder and decoder rely, so that similar sentences have close representations. When implementing POS Tags in the output of the encoder of our Transformer model, the intermediate representation should be enhanced, since additional information is added to the system. It is interesting to improve the model in such a way as it usually leads to a better translation performance.

CHAPTER III

10 Code development and implementation

The first step in the code development was to understand the baseline from which the project was started and then introducing POS tagging in the encoder. In this chapter, we present the POS Tag libraries, the baseline and we detail all the changes made to it in order to make the Part-Of-Speech tagging possible. We also explain the two different systems that we have developed and trained as well as the data used to train them and the setbacks experienced while training.

10.1 POS Tags

With the aim of doing Part-Of-Speech tagging for the source language of the transformer we have implemented the stanza[5] library, but before we implemented it, we used Spacy[6]. These two libraries have downloadable pipelines for many languages that when given a text to the NLP model of a given language, we can generate POS Tags from the text. They also have other processors apart from tagging, but this is the one that we have implemented in the encoder as the reference POS Tags from which to compare the Transformer predictions in the criterion. (Qi, Zhang, Zhang, Bolton, & Manning, 2020) (spaCy, 2016)

The problem with the Spacy library was that for Spanish and French the pipelines didn't have the POS tagging feature. Once we realized it, we proceeded to implement the Stanford NLP Group library Stanza. The Part-Of-Speech Tags used from Stanza are Universal POS Tags that mark the core Part-Of-Speech categories for all languages (English, Spanish, French and German) (UD, 2014).

• ADJ: adjective • ADP: adposition • ADV: adverb • AUX: auxiliary • CCONJ: coordinating conjunction • DET: determiner • INTJ: interjection • NOUN: noun • NUM: numeral	• PART: particle • PRON: pronoun • PROPN: proper noun • PUNCT: punctuation • SCONJ: subordinating conjunction • SYM: symbol • VERB: verb • X: other

***Table 1**:* Enumeration *of the 17 Universal Dependency Tags (UD, 2014)*

[5] https://stanfordnlp.github.io/stanza/tokenize.htm
[6] https://spacy.io/usage

10.2 Baseline and Module changes

The idea was to first train the multilingual translation baseline system (Escolano, Costa-jussa, Fonollosa, & Artetxe, 2020) with language specific encoders and decoders (Transformer model facilitated by the MT UPC research group). The baseline model was important to be trained so that later, when we implemented POS tagging to that same code, we could compare the improvements (or demeanings) in the translation task.

We made several module changes in the baseline code in order to insert the POS tagging feature in the encoder with the goal of improving the intermediate representation of the Transformer.

From the source language given to the Transformer, we generated POS Tags with Stanza. For example, if the input text was “bribery and corruption”, as in figures 3 and 4, then the Stanza POS Tags would be “PROPN CCONJ PROPN”. These were used as the golden truth to compare the POS Tag predictions and optimize the loss function. The modified Transformer has two outputs, one for the prediction of the translation and one of the predictions of the POS Tags of the source text. The second output is an array of the size of the number of all possible POS Tags used by Stanza (17) each one with a probability. From the highest probabilities, the POS Tag predictions are associated with a Universal POS label.

10.2.1 POS tagged model with the individual classifiers

We have created a first model so that each language has its own POS Tag classifier, meaning that no model weights are shared. From the plug-ins defining the Fairseq model we have modified the Model, the Task and the Criterion. The changes are described below:

- **Model:**

In order to make the POS Tag predictions, we have added an additional linear layer to the Transformer’s Encoder. This layer predicts 17 probabilities (one per UD label) for each word from the encoder’s output and since it works at word level, we had to reshape the encoder output before implementing it (see figure 3). After the linear layer, the encoder output had to be reshaped once more so that the translation prediction was at sentence level again.

- **Task:**

The task contains a function in which the dataset is built. We have implemented changes so that the language-pair dataset imports Stanza and its pipelines for all the languages. The tokenization of the source text induced some problems, since instead of words the Transformer is translating subwords. The vector containing all the Stanza POS Tags had to have the same size as all the possible subwords of the text.

- **Criterion:**

In the Criterion, we implemented the POS Tag loss. This loss is added to the translation loss and we define the sum as the final loss we want to optimize. This way translation and POS Tag prediction will be optimized jointly. In the cross-entropy loss, the predicted Tags and translations are compared with the real ones from Stanza and the Europarl human translations respectively.

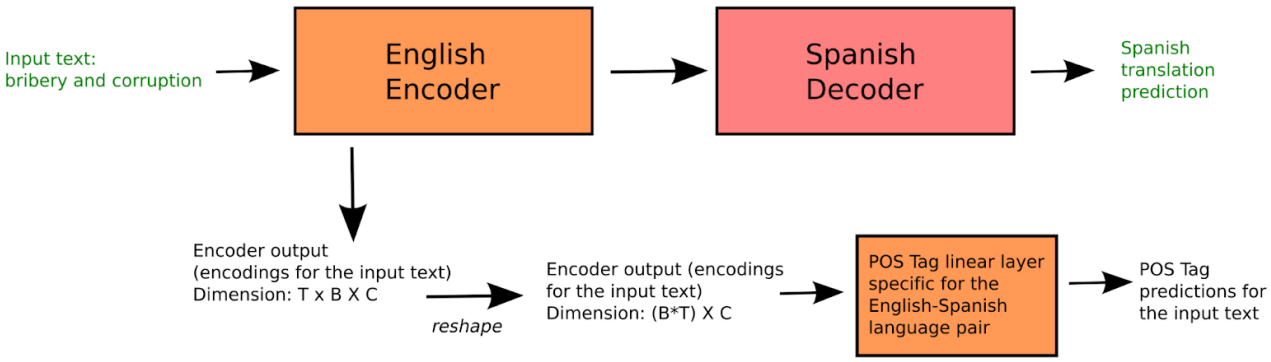

***Figure 3**: Diagram of how the model works for the English-Spanish language pair. T is the number of tokens, B the batch size and C the embedding dimension*

10.2.2 POS tagged model with the shared classifier

The aim of the second encoder POS Tag implementation is to have the same classifier for all languages to further help the intermediate space representation. In the previous case, the linear layer we introduced was different for each encoder so that it could learn specific information of each language. Now, as we share the layer, this is the same for all languages meaning that it learns more general information.

For that, with respect to the previous model, we have added a new argument to the task concerning the shared POS Tag layer. This way, when this argument is present in the training, a linear layer (of dimension 17 as the dictionary of the POS Tags) is created in the Model plug-in and used in the encoder, the same for each language pair, to make predictions as in the previous implementation. Since the layer is shared between models and they have the same weights, it should help the common intermediate representation to be more similar between languages.

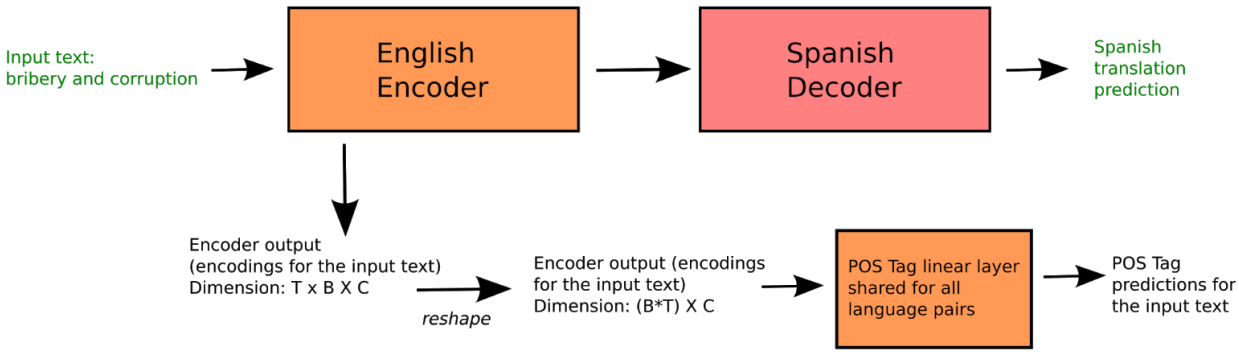

***Figure 4**: Diagram of how the model works for the English-Spanish language pair. T is the number of tokens, B the batch size and C the embedding dimension*

10.3 Data

All the models were trained using 2 million sentences in English, Spanish, French and German from the Europarl corpus (Koehn, Europarl: A Multilingual Corpus for Evaluation of Machine Translation, 2002)[7]. This corpus of parallel text including 11 languages and collected from the proceedings of the European Parliament is used in the field of Natural Language Processing and has various applications in Statistical Machine Translation (Koehn, Europarl: A Parallel Corpus for Statistical Machine Translation, 2005). It contains human-made translations of the source text for each language, which are used in the training phase of the Transformer to compare with the translation predictions.

We used newstest2012 and newstest2013 from WMT[8] as validation and test sets, the last one used in the Machine Translation task from the experimental framework.

The data fed to the system was already pre-processed and binarized according to standard Moses scripts. The Moses open-source toolkit is a translation system containing all the necessary components to pre-process data and train translation models (Koehn, et al., 2007).

10.4 Training

The Transformer architecture we used in this thesis required a powerful computer containing a GPU, that is why we carried out the training remotely on a UPC server. The baseline code was downloaded to the server, and the POS tagged ones were updated with the git commands and a GitHub account since we made the code changes locally, and then we sent them for training.

During the training and experimentation phases, we used a stack of 6 encoders and 6 decoders with 8 attention heads in the Farseq written Transformer model. We performed the trainings of the systems with a 12GB GPU, embedding size of dimension 512 and

[7] https://www.statmt.org/europarl/
[8] http://www.statmt.org/

vocabulary size of 32k subword tokens with Byte Pair Encoding. Also, the dropout was 0.3 after every layer and the Adam optimizer was used with a learning rate of 0.001 and 4000 warmup steps. In addition, we trained the model with a 0.1 label smoothing factor of the cross-entropy.

The models were trained in the 4 aforementioned languages and since the encoder and decoder languages are different this makes a total of 12 language pairs, meaning that the models were trained on all 12 translation directions simultaneously.

As output of the training, we obtained a .log file for every model in which you could see the current total valid loss and the best minimum loss. The validation loss changed every epoch, that is when the learning algorithm has worked during the entire dataset. When the valid loss was different from the best loss, we could stop the training, since that is the reference loss we take for achieving the best parameters of the model and training it further would probably mean overfitting it.

```
--arch interlingua_transformer \
--save-dir checkpoints/europarl \
--optimizer adam \
--adam-betas '(0.9, 0.98)' \
--clip-norm 0.0 \
--lr-scheduler inverse_sqrt \
--warmup-init-lr 1e-07 \
--warmup-updates 4000 \
--lr 0.001 \
--min-lr 1e-09 \
--dropout 0.3 \
--weight-decay 0.0 \
--criterion label_smoothed_cross_entropy \
--label-smoothing 0.1 \
--max-tokens 2000 \
--update-freq 16  \
--save-interval-updates 30000  \
--task interlingua_nodistance_translation   \
--lang-pairs en-de,fr-de,es-de,de-en,fr-en,es-en,en-fr,es-fr,de-fr,de-es,en-es,fr-es \
--freeze-schedule n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n,n-n \
--tie-lang-embeddings \
--adapt-schedule False \
--encoder-normalize-before \
--decoder-normalize-before \
--no-epoch-checkpoints
```

***Figure 5**: Parameters, model, optimizer, criterion, task and languages used in the training*

10.5 Setbacks during the training

Once we finished the code implementation for the first POS tagged model, we sent it to training, although several setbacks made it difficult to obtain a final functional Transformer.

The first problem, as mentioned earlier, was the Spacy POS Tag implementation as there were languages that didn't have this feature.

After the Stanza library was applied, we discovered that it tokenized again the already tokenized text and gave problems with the POS Tags assignment and dimension. Furthermore, instead of assigning POS Tags word by word, Stanza was predicting the POS Tags token by token producing the wrong real POS Tags, as the subwords (tokens) were not the whole real word and Stanza didn't recognize them right. In between, another error appeared because the source text used to implement the POS Tags had padding and the function split that we used to separate the text was eliminating that padding. We created a function so that the real source text could be obtained from the padded text and the source dictionary.

We solved the POS Tag assignment dimension errors with two matching functions and a pre-tokenized condition. Actually, we first introduced the two matching functions and then found out that they were not necessary with the pre-tokenized condition. We maintained them anyway to control any unusual tokenization case. The first function compares the word tokenization of Stanza with the real text words to obtain a POS Tag vector of the size of the number of entire words. The second function assigns the same POS Tags to the BPE subwords belonging to the same word. In order to get the complete source words from the Byte Pair Encoding and the BPE subwords, the end-word symbol '@@' had to be eliminated with a replace function. Also, when revising the model, we noticed that stanza was assigning a POS Tag to each letter, since the text was splitted one level too far and separated words in letters. Once we solved all the above, the dimensions fit.

During training we had some problems with the UPC server, first one of the machines was not working right and after the disk was so full that it cancelled some of the training models. The server cancelling models happened several times. Also, the torch library was actualized from the environment from which we executed the training and since the model worked with an old version of torch it induced some errors.

Finally, it seemed as if the model was training right and had passed a few epochs so we tried to generate a translation between two languages. The translation didn't work and we discovered another error related to the reshape made in the encoder output to implement the linear POS Tag layer. The translation prediction output was at word level and it had to be at sentence level (see section 10.2.1), which in turn produced some memory problems too. After we fixed it, the model was final.

All the above-mentioned errors meant that the training had to be restarted and caused the finalization of the training to be postponed for several days.

For the second model with the shared linear layer, we had no code setbacks, since the implementation didn't change that much, nevertheless, it was also cancelled various times by the server.

CHAPTER IV

11 Experiments in Machine Translation

For the first experiments, we generated translations between all language pairs with the trained systems and we computed the BLEU score to see the quality of the translation. Also, for the POS tagged models we visualized the POS labels to see how well the implementation worked.

11.1 Baseline model

The obtained BLEU results for the baseline system are:

en-fr	en-de	en-es	fr-en	fr-de	fr-es	de-en	de-fr	de-es	es-en	es-fr	es-de
29.64	22.11	29.83	26.42	19.55	29.36	24.68	25.83	25.15	27.80	30.22	19.96

***Table 2**: BLEU for each language pair of the baseline*

The average BLEU is 25.88.

Here are explicit examples of the translations we generated with the baseline model of the following source and target sentences:

- **English:** bribery and corruption were rife and unhindered, and the people would vote for a result which had been bought
- **Español:** el soborno y la corrupción se extendieron sin impedimento alguno y la gente acudía a las urnas con el voto apalabrado de antemano
- **Deutsch:** die Korruption wucherte hemmungslos , und das Volk ging mit bereits gekauften Stimmen zur Wahl
- **Français:** les dessous-de-table et la corruption proliféraient sans limites et le peuple se rendait aux élections avec des voix déjà vendues

We did the translation for each language pair and displayed it for a better visualization of how the system works.

Language Pair	Translation Hypothesis
en-fr	les pots @-@ de @-@ vin et la corruption étaient monnaie courante et libres , et les citoyens voteraient pour un résultat acheté
en-de	Bestechung und Korruption waren weit verbreitet und ungehindert , und die Menschen würden für ein Ergebnis stimmen , das
en-es	el soborno y la corrupción han prosperado y sin obstáculos , y los ciudadanos votarían a favor de un resultado que se había comprado

fr-en	submis- and corruption were rampant without any restrictions , and the people went to the polls with voices already sold
fr-de	die Drücke- und Korruptionsfälle wuchsen grenzenlos , und das Volk reiste mit bereits verkauften Stimmen zu den Wahlen
fr-es	las condiciones infranqueables y la corrupción proliferaban sin límites y el pueblo se dirigía a las elecciones con voces que ya habían sido vendidas
de-en	corruption was rampant , and the people voted ' by already bought @-@ out votes
de-fr	la corruption s' est développée sans limites et le peuple a voté avec des voix déjà rachetées
de-es	la corrupción se extendió sin freno y el pueblo votó con votos ya adquiridos
es-en	corruption and bribery spread unhindered and people went to the polls with the vote beforehand
es-fr	la corruption et la corruption se sont propagées sans aucune entrave et les citoyens se sont rendus aux urnes par un vote préalable
es-de	Bestechung und Korruption breiteten sich ungehindert aus , und die Mensche n gingen mit dem im Voraus verabschiedeten Votum an die Wahlurne

***Table 3**: Translations between all the language pairs generated by the baseline model*

11.2 POS tagged model with the individual classifiers

The POS tagged model with the individual classifiers gave similar results to those of the baseline model. You can see the BLEU scores below:

en-fr	**en-de**	**en-es**	**fr-en**	**fr-de**	**fr-es**	**de-en**	**de-fr**	**de-es**	**es-en**	**es-fr**	**es-de**
29.24	21.53	28.97	25.85	18.91	29.09	23.97	25.12	24.80	27.09	29.92	19.71

***Table 4**: BLEU score for the POS tagged model* with the individual classifiers

The average BLEU for this system is 25.35.
In the following we also show the translated sentences for the same references and targets as in the baseline. Comparing Table 3 and Table 5, the hypothesis translations look very alike.

Language Pair	**Translation Hypothesis**
en-fr	les pots @-@ de @-@ vin et la corruption étaient monnaie courante et les citoyens voteraient pour un résultat qui avait été acheté
en-de	Bestechung und Korruption waren weit verbreitet und ungehindert , und die Menschen würden für ein Ergebnis stimmen
en-es	el soborno y la corrupción proliferaban y sin obstáculos , y el pueblo votaría un resultado que se había comprado

fr-en	the desider- and corruption were rampant , and the people went to the elections with votes already sold
fr-de	die Unwägbarkeiten und die Korruption breiteten sich ohne Grenzen aus , und die Menschen gingen mit bereits verkauften Stimmen in die Wahlen
fr-es	los desiderata y la corrupción proliferaban sin límites y el pueblo se dirigía a las elecciones con voces ya vendidas
de-en	corruption was rampant , and the people voted
de-fr	la corruption s' est déchaînée et le peuple a voté
de-es	la corrupción se ha propagado sin trabas y el pueblo ha votado
es-en	bribery and corruption spread unhindered and people went to the polls with a vote in advance
es-fr	les pots @-@ de @-@ vin et la corruption se sont étendus sans entrave et les électeurs se sont rendus aux urnes par le vote dit à l' avance
es-de	Bestechungsgelder und Korruption haben sich ungehindert ausgebreitet , und die Menschen haben die Wahlurnen mit der vorgeschobenen Abstimmung

***Table 5**: Translations between all the language pairs*

Although sentences are similar in Tables 3 and 5, for the German-English and German-French language pairs the model did not finish the translation, which causes the BLEU score to decrease. When looking to other source sentences of model we could conclude that this was not usually the case.

Apart of showing the translation predictions, we also retrieved the POS Tags to see how well the system implemented the word predictions. When visualizing the POS Tag prediction, we saw that it most usually predicts nouns, proper nouns and unknown labels. Also, the prediction probabilities were all very low, there was no probability significantly bigger than the others, meaning that the system seems no to have learned much. However, between language pairs where the source language is the same, the results were very similar, which makes sense since the POS Tag prediction is implemented in the encoder for the source text.

11.3 POS tagged model with the shared classifier

For the model in which the POS Tag prediction linear layer is shared between all the language pairs, BLEU results are very similar to those of the baseline too.

en-fr	en-de	en-es	fr-en	fr-de	fr-es	de-en	de-fr	de-es	es-en	es-fr	es-de
28.94	21.21	28.78	26.21	18.72	28.63	24.16	24.91	24.73	27.02	29.75	19.33

***Table 6**: BLEU score for the POS tagged model with the shared classifier*

The average BLEU is 25.20 in this case.

Below we show once more the examples of translation for each language pair.

Language Pair	Translation Hypothesis
en-fr	la corruption et la corruption étaient monnaie courante et libres , et le peuple voterait pour un résultat qui avait été acheté
en-de	Bestechung und Korruption waren weit verbreitet und ungehindert , und das Volk würde für ein Ergebnis stimmen , das gekauft worden war
en-es	el soborno y la corrupción fueron frecuentes y libres , y los ciudadanos votarían a favor de un resultado que se había comprado
fr-en	the people went to the polls with voices already sold
fr-de	die Dekabel und die Korruption breiten sich ohne Grenzen aus , und das Volk ging mit bereits verkauften Stimmen zu den Wahlen , mit Stimmen
fr-es	las personas que estaban fuera de lugar y la corrupción proliferaban sin límites , y el pueblo acudía a las elecciones con voces ya vendidas
de-en	corruption was rampant ,and the people were already voting with loud voices to vote
de-fr	la corruption s' est répandue sans entrave et le peuple a voté
de-es	la corrupción se intensificó sin trabas , y el pueblo ya había votado
es-en	bribery and corruption went on unhindered , and people went to the polls with the premeditated vote
es-fr	la corruption et la corruption se sont répandues sans aucune entrave et les gens se sont rendus aux urnes en votant à l' avance
es-de	Bestechung und Korruption haben sich ungehindert ausgebreitet , und die Menschen gingen mit der vorausgegangenen Abstimmung zu den Wahlurnen

***Table 7**: Translations between all the language pairs*

For this model the POS Tag prediction results were almost the same as in the model above. One reason the POS Tagged models do not predict POS Tags with high precision could be that the implemented classifier is too simple and does not predict correctly. We wanted a very simple classifier, which would not disturb the translation task, with the ultimate goal of enhancing the intermediate representation and improving the quality of the translation.

12 Experiments in Natural Language Inference

Natural Language Inference is the task of determining, given a natural language hypothesis (h) and a natural language premise (p), whether the former can reasonably be inferred from the latter. The problem aims to deduce if the hypothesis and the premise contain a relationship of entailment, contradiction or neither (neutral) (MacCartney, 2009). It is also known as Recognizing Textual Entailment (RTE) and it is a practical method for testing on sentence comprehension (Alexis Conneau, 2018).

The purpose of implementing NLI in this work is to compare and study the intermediate space that we previously trained in all configurations of the Transformer multilingual machine translation system. The three architectures we trained are the baseline and the two POS tagged models. What we did was train the systems in one language for the NLI task, English, and then evaluate them in other languages applying cross-lingua understanding. A classifier fed with the encoding of the reference and the hypothesis sentences was trained using the English encoder and we measured the performance for all the languages with the accuracy score. (Escolano, Costa-jussa, Fonollosa, & Artetxe, 2020)

12.1 Data and training

The training data we used for the NLI task was the Multi-Genre Natural Language Inference (MultiNLI) corpus[9] composed by 433k sentence pairs with annotation of textual entailment. (Williams, Nangia, & Bowman, 2018)

We used the Cross-lingual Natural Language Inference (XNLI) corpus[10] as test and validation sets with 5.000 test and 2.500 development examples (Conneau, et al., 2018). This corpus is extended in 15 languages, which are English, French, Spanish, German, Greek, Bulgarian, Russian, Turkish, Arabic, Vietnamese, Thai, Chinese, Hindi, Swahili and Urdu. XNLI is a benchmark for cross-lingual sentence encoding, allowing to evaluate how to perform inference in any of the 15 languages when only NLI data for English is available in the training. (Conneau, et al., 2018)

The English encoder of the English-Spanish pair trained with Europarl (section 8) for the three aforementioned architectures was the one we employed for the NLI training and task. We trained it with the 128 hidden units classifier and then used it in cross-lingua understanding.

[9] https://cims.nyu.edu/~sbowman/multinli/
[10] https://cims.nyu.edu/~sbowman/xnli/

12.2 Baseline model

After the NLI training of the English encoder we evaluated Natural Language Inference for all languages.

en	0.523
de	0.492
es	0.497
fr	0.503

Table 8: Accuracy for each language in the NLI task of the baseline

Table 10 shows the accuracy results for the baseline model. The metric's scores are quite good, meaning that the baseline already has a coherent intermediate representation. The average NLI accuracy is 0.504.

12.3 POS tagged model with the individual classifiers

The results we obtained in the NLI task for the model in which each language has its own POS Tag classifier are the following:

en	0.410
de	0.405
es	0.420
fr	0.392

Table 9: Accuracy for each language in the NLI task of the POS tagged model with individual classifiers

From the table above, we can infer that the accuracy results are significantly worse than in the baseline. This is probably due to the problems mentioned in the section of Machine Translation experiments (section 9). For this case, the average accuracy is 0.407, almost 0.1 points lower than in the case of the baseline.

12.4 POS tagged model with the shared classifier

The Natural Language Inference performance for the model that has the same POS Tag classifier for all languages is better than the model above, but still worse than the baseline.

en	0.465
de	0.450
es	0.444
fr	0.424

Table 10: Accuracy for each language in the NLI task of the POS tagged model with the shared classifier

As can be computed from table 12, the average accuracy is 0.445.

13 Visualization of intermediate space representations

For this section, we used the Visualization tool of intermediate representations from (Escolano, Costa-jussa, Lacroux, & Vazquez, 2019) and freely available in GitHub[11]. This website application, implemented with the Bokeh and Flask python libraries, allows layer representations to be visualized at sentence and word level. Besides visualizing the intermediate representations of source sentences and their words, it also helps visualize the evolution of the embeddings within all the layers of the decoder.

The input data required for the tool are the encoding of sentences, the embedding of words and the text sentences of the visualized model. In order to get all this information, we retrieved encodings and embeddings from the checkpoint documents saved during the models training (section 8) and we recovered the text sentences from the encodings using the language dictionary too. Applying the dimensionality reduction technique UAMP[12] to the input, the sentence and word representations are plotted in two dimensions maintaining relations between vectors. The tool can be applied to either monolingual or multilingual systems and to one layer or multi-layer intermediate representations. (Escolano, Costa-jussa, Lacroux, & Vazquez, 2019)

The tool works with a single json file where all the information required for the visualization is present. That is why, during the dimensionality reduction, we modelled the input so that in a json document the sentences, its representations and the tokens embeddings were displayed following the format below:

```
{
    "content": {
        "id": {
            "sentence": "",
            "embedding": "",
            "words": {
                "word1":"",
                "wordN":""
            }

        }
    }

}
```

Figure 6: JSON structure

What we did was use the encoder's output as input to the tool, previously arranged as in figure 6, and then we visualized words and intermediate sentence representations.

[11] https://github.com/elorala/interlingua-visualization
[12] https://www.theoj.org/joss-papers/joss.00861/10.21105.joss.00861.pdf

13.1 Baseline model

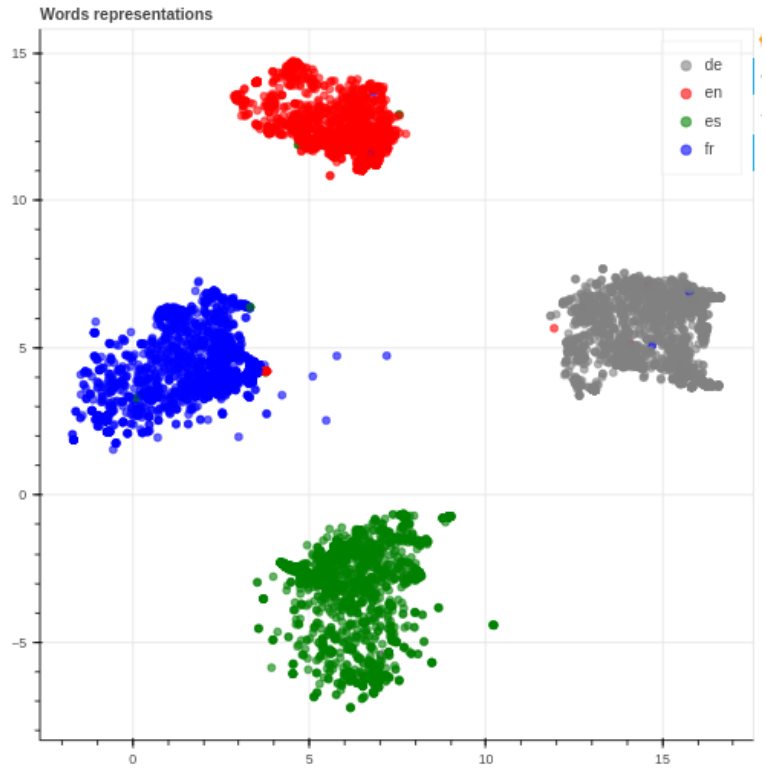

***Figure 7**:* Word embeddings of the baseline

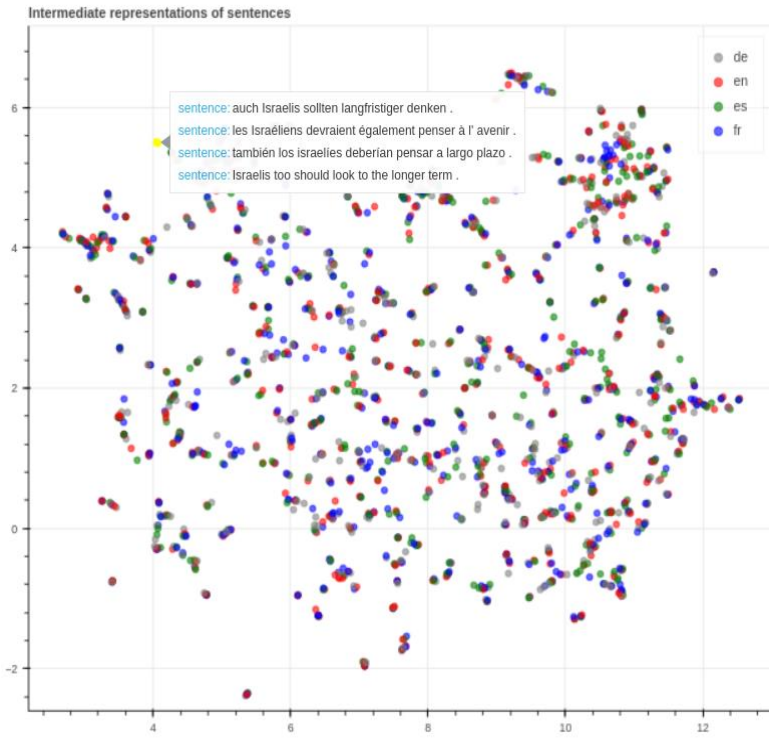

***Figure 8**:* Sentence encodings of the baseline

As can be seen in the figures above, the words embeddings for each language are placed in separate clusters, one for each language, contrary to the sentence encodings. In figure 8, sentences in different languages having the same meaning, being each other's translations, are placed almost in the exact same point. This fact leads one to think that the baseline model already has a coherent intermediate space representation of sentences for the 4 languages in which we have trained it.

13.2 POS tagged model with the individual classifiers

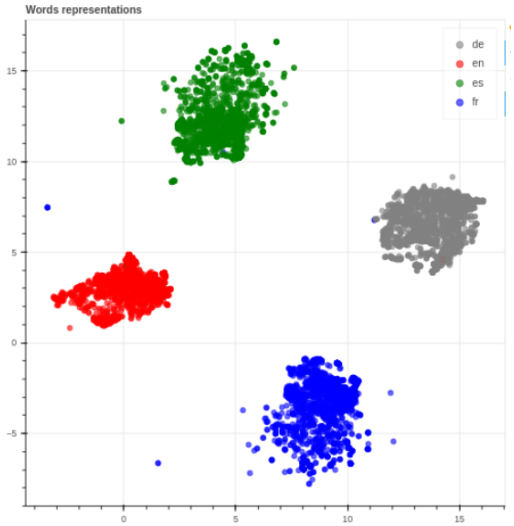

***Figure 9**: Word embeddings for the POS tagged model with the individual classifiers*

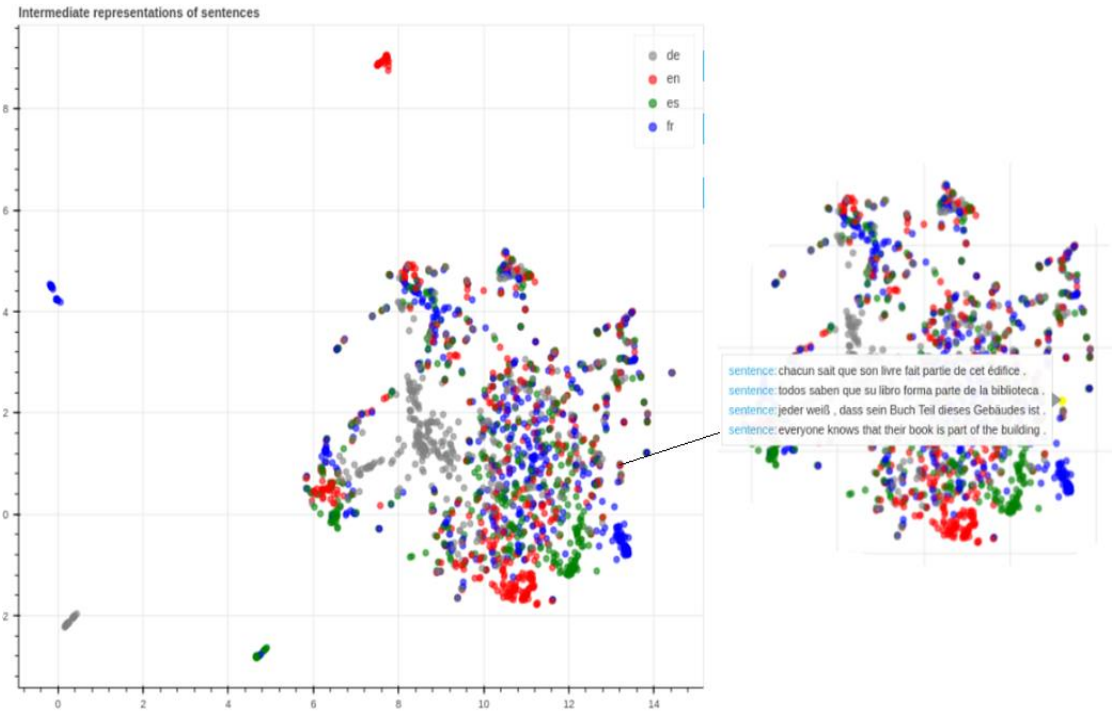

***Figure 10**: Sentence encodings for the POS tagged model with the individual classifiers*

Note that in figure 9 the words embeddings are still separated in different clusters as in the baseline, but the sentence encodings have worsened due to the POS tag implementation (figure 10). There are sentences with the same meaning in different languages that are represented in the same spot again, but there are others that are not. For example, in figure 10, we can see how for German there are several sentences represented in a separate cluster away from the other languages.

13.3 POS tagged model with the shared classifier

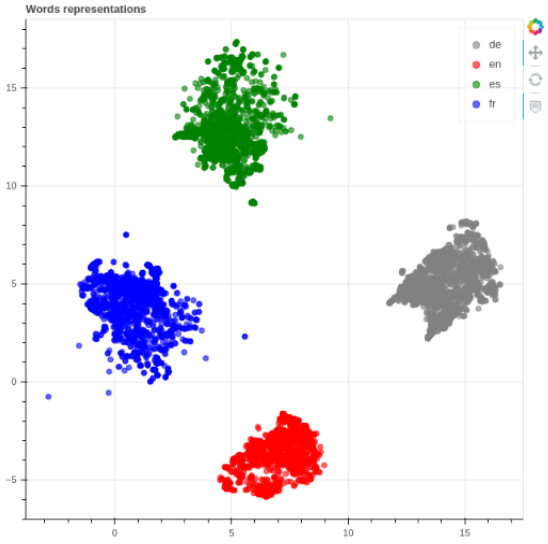

***Figure 11**: Word embeddings for the POS tagged model with the shared classifier*

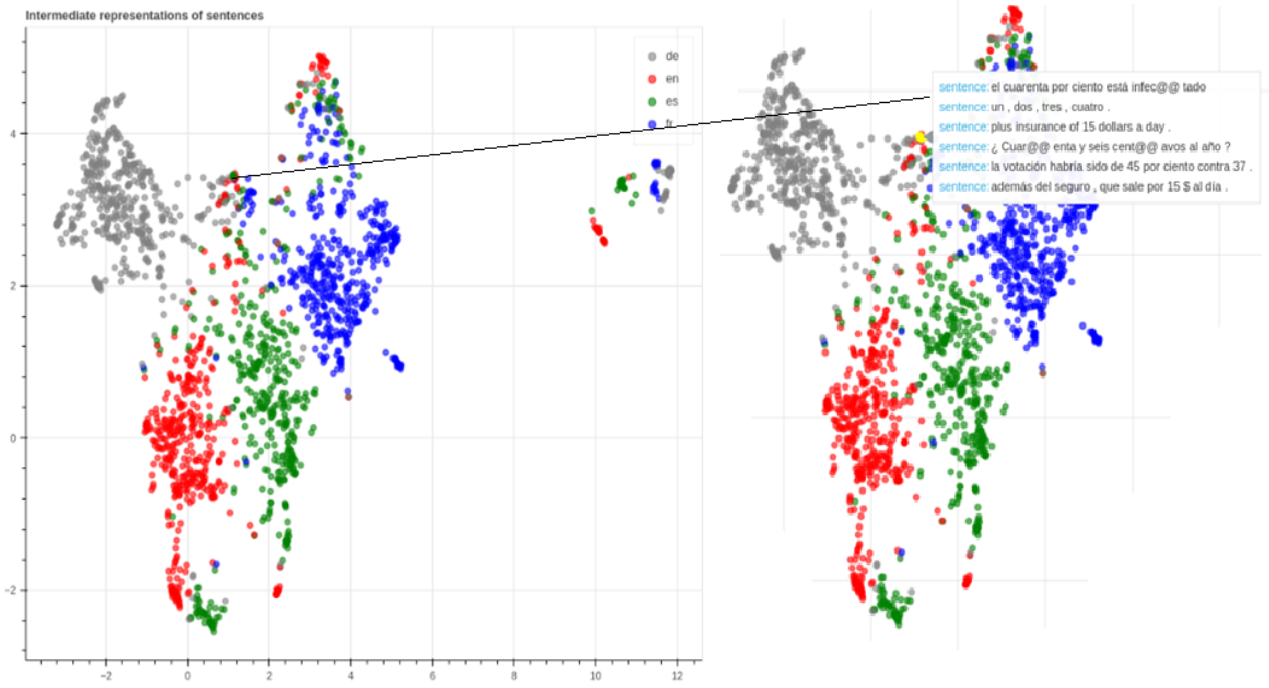

***Figure 12**: Sentence encodings for the POS tagged model with the shared classifier*

In the case of the model with the shared classifier between languages, the sentence encodings are worse. In figure 12 we see how the sentences are represented separately by language regardless of their meaning.

14 Source embeddings experiment

The goal of this experiment is to measure the effect of the POS Tag implementation on the contextual representation of the source tokens created by the encoder. For that we will study how source embeddings codify gender information as has been done previously in (Costa-jussà, et al., 2020). Even though Part-Of-Speech Tags do not have specific gender information, implementing them in the encoder and adding extra information about the grammatical category may help improve gender prediction.

Using WinoMT as the data set, specialized in evaluating gender bias, we chose two word types for English, determiners and professions, to study gender information at their contextual embeddings. Since determiners in English have no gender information, their information will come from the sentence context. (Costa-jussà, et al., 2020)

The data set used for training and testing, as mentioned before, is WinoMT (Stanovsky, Noah, & Zettlemoyer, 2019). This is a set concentrated on gender bias in MT containing 38888 sentences, which on one side is distributed between male, female and neutral sentences, and on the other hand between stereotypical, anti-stereotypical and neutral gender-role assignments.

14.1 Training and testing

What we did was train and test a classifier to see how well the gender prediction worked for determiners and occupations taking English as the source language and using its embeddings. For the three trained models of language specific encoder and decoder (section8), we trained a support-vector network (SVM), which is a learning machine for two-group classification problems (CORTES & VAPNIK, 1995), with 1000 sentences from WinoMT chosen randomly and then we test it with the rest of the set sentences.
We performed the test 5 times for each of the models, since there is a random factor in the token representation. The higher the accuracy of the prediction, the more information about gender will be encoded in the source embeddings.

14.2 Results

Determiners			Profession		
Baseline model	**POS tagged model with the individual classifiers**	**POS tagged model with the shared classifier**	**Baseline model**	**POS tagged model with the individual classifiers**	**POS tagged model with the shared classifier**
0.647	0.639	0.619	0.699	0.720	0.706
0.651	0.638	0.610	0.705	0.704	0.698
0.631	0.635	0.625	0.691	0.717	0.706
0.626	0.663	0.626	0.709	0.713	0.714
0.659	0.656	0.633	0.698	0.717	0.723
Average					
0.643	0.646	0.622	0.701	0.714	0.710

***Table 11**: Accuracy results for determiners and professions*

As can be seen in table 11, we used accuracy as the measuring score. From the results of the table above, we did a bar graph for better visualization:

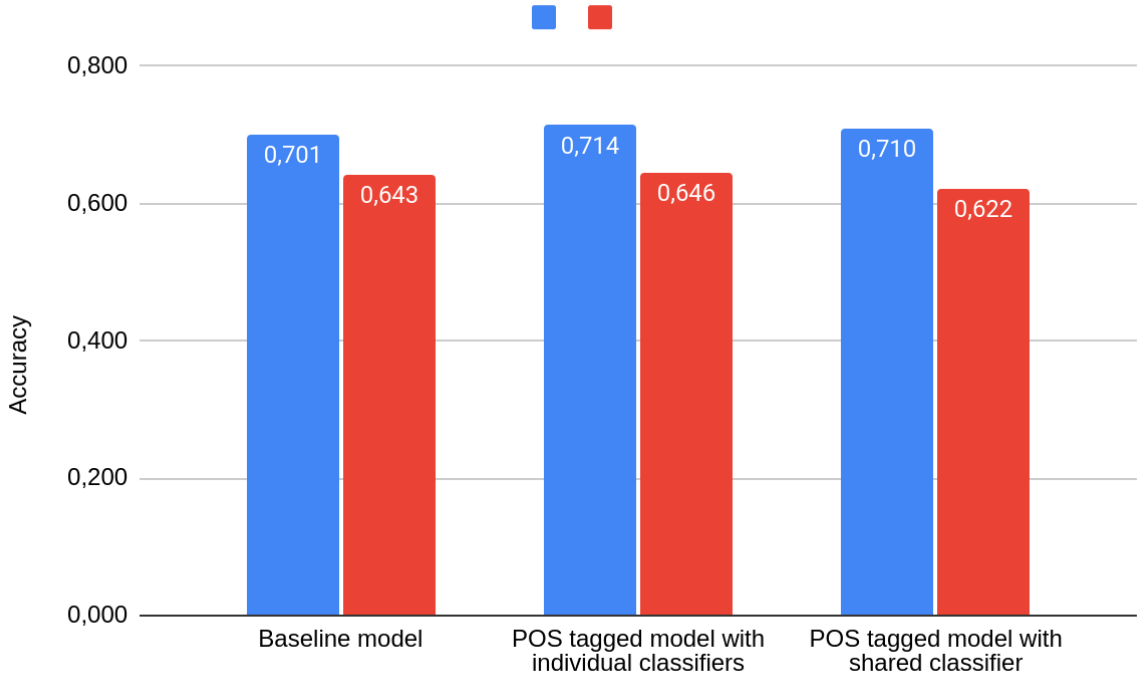

***Figure 13**: Determiner predictions in red and Profession predictions in blue*

From figure 13, we can see how the prediction for professions has increased 0.1 points in the two POS tagged models with respect to the baseline. For the case of the determiners, the accuracy has only increased a bit in the POS tagged model with the individual layers.

Tables 12 and 13 from the appendix show the 50 most common wrongly classified determiners and professions, and almost all of them are repeated for every model. In the case of the determiners, examples of the most misclassified ones are receptionist, librarian, baker and cashier. Examples for the professions are nurse, tailor and cleaner. All of these professions have a stereotypical gender assigned in society and when the gender is inverted the classifier does not predict correctly. This happens because normally, the training data is already biased, as it is a recompilation of sentences made by humans.

CHAPTER V

15 Conclusions

Taking everything into consideration, it can be concluded that the Part-Of-Speech Tag classifier introduced to the two models we have developed during the bachelor thesis does not help fulfil all the hypotheses that we had at the beginning. Nevertheless, it improves the gender determination of Determiners and Professions for the encoder contextual embeddings and we found new ideas along the way.

The translation BLEU score has not decreased very much, meaning that the system still works and makes good translations. The same happens for the NLI task, in which the performance of the POS tagged models is not that different to the performance of the baseline.

Actually, the model with the shared POS Tag layer performs better in NLI than the model with the individual classifiers for each language, but it seems to have a worst intermediate representation. Since the intermediate representation visualization is very complex and involves dimensionality reduction, it is possible that there is some connection lost we cannot perceive.

In the case of the common intermediate representation visualization, it is clear how implementing POS labels in the encoder of the Transformer model affects the space representation. The results show how the systems learn more language-specific representations.

Some of the reasons that may have contributed to not fulfil all the initial hypothesis are:

- The best validation loss does not translate in the improvement of the BLEU score or the POS Tag prediction. This is due to the fact that, in the case of translation, the probability of the right tokens can improve generally, but you can also commit more errors. Furthermore, as in the POS tagged models two losses are added, the POS Tag prediction can be better but not translation and otherwise, making the model not be correctly trained for both features.
- As it appears in the Machine Translation task section, the POS Tag prediction precision is not very high, probably because the implementation of the POS Tags in the encoder was too simple. Still, the output of the predicted POS Tags is not random and it predicts a lot of nouns, proper nouns and labels it does not recognize, which are the most common POS Tags in a text.

16 References

Alammar, J. (2018). *The Illustrated Transformer*. Retrieved from http://jalammar.github.io/illustrated-transformer/

Alammar, J. (2018). *Visualizing A Neural Machine Translation Model*. Retrieved from https://jalammar.github.io/visualizing-neural-machine-translation-mechanics-of-seq2seq-models-with-attention/

Amato, B., Durocher, A., Hurtado, G., Jouandin, A., & Marois, V. (2019). *Learning about the Attention Mechanism and the Transformer Model.*

Aprendizaje Automático. (2020). Retrieved from Clasificación: Exactitud: https://developers.google.com/machine-learning/crash-course/classification/accuracy

Arivazhagan, N., Bapna, A., Firat, O., Lepikhin, D., Johnson, M., Krikun, M., . . . Wu, Y. (2019). *Massively Multilingual Neural Machine Translation.* Google AI. arXiv.

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural Machine Translation by Jointly Learning to Align and Translate.* Jacobs University Bremen; Université de Montréal. Bremen, Montréal: arXiv.

Conneau, A., Lample, G., Rinott, R., Schwenk, H., Stoyanov, V., Williams, A., & Bowman, S. (2018). *The Cross-Lingual NLI Corpus (XNLI).*

Conneau, A., Rinott, R., Lample, G., Schwenk, H., Stoyanov, V., Williams, A., & Bowman, S. R. (2018). *XNLI: Evaluating Cross-lingual Sentence Representations.* Facebook AI, New York University.

CORTES, C., & VAPNIK, V. (1995). *Support-Vector Networks.*

Costa-jussà, M. R., Escolano, C., Basta, C., Ferrando, J., Batlle, R., & Kharitonova, K. (2020). *Gender Bias in Multilingual Neural Machine Translation: The Architecture Matters.* TALP Research Center, Universitat Politècnica de Catalunya. Barcelona: arXiv.

Escolano, C. (2018). *Interlingua based Neural Machine Translation.* Barcelona: UPC.

Escolano, C., Costa-jussà, M. R., Fonollosa, J. A., & Artetxe, M. (2020). *Multilingual Machine Translation: Closing the Gap between Shared and Language-specific Encoder-Decoders.* TALP Research Center, Universitat Politècnica de Catalunya; IXA NLP Group, University of the Basque Country. arXiv.

Escolano, C., Costa-jussa, M. R., Fonollosa, J. A., & Artetxe, M. (2020). *Training Multilingual Machine Translation by Alternately Freezing Language-Specific Encoders-Decoders.* TALP Research Center (UPC), IXA NLP Group (UPV/EHU). arXiv.

Escolano, C., Costa-jussa, M. R., Lacroux, E., & Vazquez, P.-P. (2019). *Multilingual, Multi-scale and Multi-layer Visualization of Intermediate Representations.* Universitat Politecnica de Catalunya. Barcelona: arXiv.

Facebook AI, R. (2018). *fairseq*. Retrieved from fairseq documentation: https://fairseq.readthedocs.io/en/latest/

Gu, J., Wang, Y., Cho, K., & Li, V. O. (2019). *Improved Zero-shot Neural Machine Translation via Ignoring Spurious Correlations.* Facebook AI Research; The University of Hong Kong; New York University, CIFAR Azrieli Global Scholar. arXiv.

Gupta, T. (2017). *towards data science*. Retrieved from Deep Learning: Back Propagation: https://towardsdatascience.com/back-propagation-414ec0043d7#.tje3h7wi0

Gupta, T. (2017). *towards data science*. Retrieved from Deep Learning: Feedforward Neural Network: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7

JournalDev. (2010). *JournalDev*. Retrieved from What is Pytorch?: https://www.journaldev.com/35641/what-is-pytorch

Kingma, D. P., & Ba, J. L. (2017). *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION.* arXiv.

Koehn, P. (2002). *Europarl: A Multilingual Corpus for Evaluation of Machine Translation.* Information Sciences Institute, University of Southern California.

Koehn, P. (2005). *Europarl: A Parallel Corpus for Statistical Machine Translation.* University of Edinburgh, School of Informatics.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., . . . Herbst, E. (2007). *Moses: Open Source Toolkit for Statistical Machine Translation.* aclweb.

Luong, M.-T., Pham, H., & D.Manning, C. (2015). *Effective Approaches to Attention-based Neural Machine Translation.* Stanford University, Computer Science Department. Stanford: arXiv.

MacCartney, B. (2009). *NATURAL LANGUAGE INFERENCE.* The Stanford NLP Group.

Mercedes Garcıa-Martınez, L. B. (2016). *Factored Neural Machine Translation Architectures.* LIUM, University of Le Mans, France.

Müller, R., Kornblith, S., & Hinton, G. (2020). *When Does Label Smoothing Help?* Google Brain. Toronto: arXiv.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). *BLEU: a Method for Automatic Evaluation of Machine Translation.* IBM T. J. Watson Research Center. Yorktown Heights, NY: Association for Computational Linguistics.

Peris, Á., Domingo, M., & Casacuberta, F. (2016). *Interactive Neural Machine Translation.* Universitat Politècnica de València, Pattern Recognition and Human Language Technology Research Center. València: ResearchGate.

PyTorch. (2021). *PyTorch*. Retrieved from DEEP LEARNING WITH PYTORCH: A 60 MINUTE BLITZ: https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Qi, P., Zhang, Y., Zhang, Y., Bolton, J., & Manning, C. D. (2020). *Stanza : A Python Natural Language Processing Toolkit for Many Human Languages.* Stanford University. Stanford: arXiv.

Sennrich, R., Haddow, B., & Birch, A. (2016). *Neural Machine Translation of Rare Words with Subword Units.* University of Edinburgh, School of Informatics. Edinburgh: arXiv.

SketchEngine. (2018). *sketchengine*. Retrieved from POS tags: https://www.sketchengine.eu/blog/pos-tags/

spaCy. (2016). *spaCy*. Retrieved from spaCy 101: Everything you need to know: https://spacy.io/usage/spacy-101

Stanovsky, G., N. A., & Zettlemoyer, L. (2019). *Evaluating Gender Bias in Machine Translation.* Seattle.

Tagliaferri, L. (2017, September 28). *DigitalOcean*. Retrieved from An Introduction to Machine Learning: https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning

UD. (2014). *Universal Dependencies*. Retrieved from Universal POS tags: https://universaldependencies.org/u/pos/

Uszkoreit, J. (2017). *Transformer: A Novel Neural Network Architecture for Language Understanding.* Google AL Blog.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . . . Polosukhin, I. (2017). *Attention Is All You Need.* arXiv.

Wang, S.-C. (2003). *Artificial Neural Network.* springer.

Williams, A., Nangia, N., & Bowman, S. (2018). *A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference.* NYU. Association for Computational Linguistics.

Yse, D. L. (2019). *towards data science*. Retrieved from Your Guide to Natural Language Processing (NLP): https://towardsdatascience.com/your-guide-to-natural-language-processing-nlp-48ea2511f6e1

17 Appendix: Misclassification List

Order	Baseline mode	POS tagged model with the individual classifiers	POS tagged model with the shared classifier
1	someone	someone	someone
2	receptionist	receptionist	salesperson
3	librarian	librarian	cashier
4	hairdresser	cashier	physician
5	baker	baker	mechanic
6	tailor	tailor	tailor
7	mechanic	counselor	receptionist
8	analyst	nurse	guard
9	cashier	analyst	carpenter
10	salesperson	laborer	clerk
11	nurse	carpenter	housekeeper
12	laborer	construction	analyst
13	counselor	mechanic	baker
14	carpenter	housekeeper	hairdresser
15	guard	cook	librarian
16	physician	hairdresser	sheriff
17	developer	salesperson	construction
18	Someone	mover	janitor
19	cleaner	cleaner	developer
20	housekeeper	janitor	cleaner
21	sheriff	sheriff	laborer
22	construction	clerk	counselor
23	designer	designer	CEO
24	CEO	Someone	attendant
25	attendant	physician	designer
26	patient	developer	editor
27	child	guard	patient
28	customer	customer	Someone
29	janitor	patient	customer

30	homeowner	child	nurse
31	clerk	client	cook
32	technician	student	child
33	student	attendant	client
34	educator	technician	accountant
35	administrator	educator	student
36	pharmacist	homeowner	homeowner
37	bartender	therapist	farmer
38	instructor	undergraduate	mover
39	paramedic	pharmacist	auditor
40	examiner	psychologist	technician
41	nutritionist	bartender	taxpayer
42	client	electrician	visitor
43	dispatcher	pathologist	electrician
44	dietitian	plumber	victim
45	chef	surgeon	instructor
46	firefighter	paramedic	examiner
47	supervisor	examiner	chemist
48	engineer	chemist	appraiser
49	therapist	nutritionist	nutritionist
50	psychologist	hygienist	programmer

***Table 12**: List of the 50 most common wrongly classified determiners*

Order	**Baseline mode**	**POS tagged model with individual classifiers**	**POS tagged model with the shared classifier**
1	someone	someone	someone
2	tailor	nurse	tailor
3	hairdresser	counselor	housekeeper
4	nurse	tailor	nurse
5	analyst	cleaner	sheriff
6	cashier	cashier	Someone
7	counselor	cook	cleaner
8	carpenter	mechanic	attendant

9	mechanic	analyst	developer
10	Someone	housekeeper	hairdresser
11	cleaner	hairdresser	guard
12	guard	clerk	baker
13	developer	carpenter	analyst
14	sheriff	guard	physician
15	salesperson	sheriff	receptionist
16	librarian	Someone	cook
17	receptionist	developer	mechanic
18	cook	librarian	librarian
19	attendant	receptionist	editor
20	baker	salesperson	salesperson
21	laborer	baker	farmer
22	clerk	supervisor	carpenter
23	supervisor	designer	clerk
24	housekeeper	customer	counselor
25	designer	patient	cashier
26	patient	mover	auditor
27	physician	child	accountant
28	customer	student	customer
29	accountant	attendant	patient
30	chief	janitor	mover
31	student	physician	child
32	child	technician	supervisor
33	mover	homeowner	driver
34	janitor	bartender	client
35	client	electrician	designer
36	administrator	plumber	homeowner
37	bartender	surgeon	student
38	victim	paramedic	janitor
39	practitioner	examiner	laborer
40	veterinarian	nutritionist	chief
41	paramedic	chef	technician
42	examiner	taxpayer	taxpayer

43	nutritionist	undergraduate	visitor
44	dietitian	pharmacist	examiner
45	CEO	witness	appraiser
46	taxpayer	officer	nutritionist
47	homeowner	appraiser	programmer
48	therapist	programmer	chef
49	visitor	paralegal	engineer
50	pharmacist	hygienist	therapist

***Table 13**: List of the 50 most common wrongly classified professions*