Multinational Address Parsing: A Zero-Shot Evaluation

Marouane Yassine, David Beauchemin, François Laviolette, Luc Lamontagne
Department of Computer Science and Software Engineering, Laval University
Group for Research in Artificial Intelligence of Laval University (GRAIL)
Québec, Canada
marouane.yassine.1@ulaval.ca, david.beauchemin.5@ulaval.ca, francois.laviolette@ift.ulaval.ca, luc.lamontagne@ift.ulaval.ca

***Abstract*—Address parsing consists of identifying the segments that make up an address, such as a street name or a postal code. Because of its importance for tasks like record linkage, address parsing has been approached with many techniques, the latest relying on neural networks. While these models yield notable results, previous work on neural networks has only focused to parse addresses from a single source country. We propose in this paper an approach in which we employ subword embeddings and a Recurrent Neural Network architecture to build a single model capable of learning to parse addresses from multiple countries at the same time while taking into account the difference in languages and address formatting systems. The proposed method achieves an average accuracy (token-wise) of 99 % on the test set of the countries used as the source dataset with no pre-processing nor post-processing being required. We explore the possibility of transferring the address parsing knowledge acquired by training on some countries' addresses to others with no further training in a zero-shot transfer learning setting. We also experiment using an attention mechanism and a domain adversarial training algorithm in the same zero-shot transfer setting to improve performance. Both methods yield state-of-the-art performance for the majority of the tested countries while giving good results on the remaining countries. We also explore the effect of incomplete addresses on our best model, and we evaluate the impact of using incomplete addresses during training. In addition, we propose an open-source Python implementation of some of our trained models[1].**

***Index Terms*—Address Parsing, Sequence labelling, Deep Learning, Zero-shot Learning, Attention Mechanisms, Domain Adversarial**

I. Introduction

Address Parsing is the task of decomposing an address into the different components it is made of. This task is an essential part of many applications, such as geocoding and record linkage. Indeed, to find a particular location based on textual data, it is quite useful to detect the different parts of an address to make an informed decision. Similarly, comparing two addresses to decide whether two or more database entries refer to the same entity can prove to be quite difficult and prone to errors if based on methods such as edit distance algorithms given the various address writing standards.

There have been many efforts to solve the address parsing problem. From rule-based techniques [1] to probabilistic approaches and neural network models [2], a lot of progress has been made in reaching an accurate segmentation of addresses. These previous pieces of work did a remarkable job at finding solutions for the challenges related to the address parsing task. However, most of these approaches either do not take into account parsing addresses from different countries or do so but at the cost of a considerable amount of meta-data and elaborate data pre-processing pipelines [3]–[6].

Our work comes with three contributions. First, we propose an approach for multinational address parsing using a Recurrent Neural Network (RNN) architecture. We start by addressing the multilingual aspect of the problem by employing multilingual sub-word units. Then we train an architecture composed of an embedding layer followed by a sequence-to-sequence (Seq2Seq) model. Secondly, we evaluate the degree to which a model trained on countries' addresses data can perform well at parsing addresses from other countries. Finally, we evaluate the performance of our models on incomplete addresses and propose a method to improve their accuracy.

II. Related Work

Since address parsing is a sequence tagging task, it has been tackled using probabilistic methods mainly based on Hidden Markov Models (HMM) and Conditional Random Fields (CRF) [2], [4], [5]. For instance, [4] proposed a large scale HMM-based parsing technique capable of segmenting a large number of addresses whilst being robust to possible irregularities in the input data. In addition, [5] implemented a discriminative model using a linear-chain CRF coupled with a learned Stochastic Regular Grammar (SRG). This approach allowed the authors to address the complexity of the features better while capturing higher-level dependencies by applying the SRG on the CRF outputs as a score function, thus taking into account the possible lack of features for a particular token in a lexicon-based model. These probabilistic methods usually rely on structured data as well as some sort of prior knowledge of this data for feature extraction or in order to implement algorithms such as Viterbi [7], especially in the case of generative methods.

In recent years, new methods [2], [3] utilizing the power of neural networks have been proposed as solutions for the address parsing problem. Using a single hidden layer feed-forward model, [6] achieved good performance. However,

This research was supported by the Natural Sciences and Engineering Research Council of Canada (IRCPJ 529529-17) and a Canadian insurance company.

[1]https://github.com/GRAAL-Research/deepparse

their approach relied on a pre-processing and post-processing pipeline to deal with the different structures of address writing and the possible prediction errors. For instance, the input data is normalized to reduce noise and standardize the many variations that can refer to the same word, such as *road* and *rd*. In addition, the model's predictions are put through a rule-based validation step to make sure that they fit known patterns. In contrast, [3] proposed a deep learning approach based on the use of RNN and did not use any pre or post-processing. Their experiments focused on comparing the performance of both unidirectional and bidirectional vanilla RNN and Long-Short Term Memory Models (LSTM) [8], as well as a Seq2Seq. The models achieved high accuracy on test sets with the Seq2Seq leading the scoreboard on most of them with no particular pre-processing needed during the inference process.

Despite reaching notable performances, the aforementioned approaches are limited to parsing addresses from a single country and would need to be adjusted to support a multinational scope of address parsing. To tackle this problem, Libpostal[2], a library for international address parsing, has been proposed. This library uses a CRF-based model trained with an averaged Perceptron for scalability. The model was trained on data from each country in the world and was able to achieve a 99.45 % full parse accuracy[3], thus defining a new state-of-the-art[4]. However, this requires putting addresses through a heavy pre-processing pipeline before feeding them to the prediction model. It is our understanding that no neural network approaches were proposed for multinational address parsing with a single model. This work aims to build a single model solution capable of parsing addresses from multiple countries, exploring the possibility of zero-shot transfer from some countries' addresses to others' and exploring (and improving) the performance on incomplete addresses.

III. Subword Embeddings

The use of subword embeddings has become popular across Natural Language Processing tasks given the performance enhancements they provide to neural network models. Word embeddings [10], [11] are usually augmented by character-level or subword-level information before being fed to the model as inputs, thus granting it a more meaningful representation of words. This strategy is employed by the word embeddings library fastText [12]. A representation of words as character n-grams is used along with word representations to produce embeddings. This approach allows for a model capable of producing richer embeddings, as well as embeddings for out-of-vocabulary words (OOV), which are computed as the sum of their n-gram fractions' embeddings. For example, the embedding of the OOV word "H1A 1B1" using a bigram model is the sum of the fractions' embedding of {H1, 1A, A1, 1B, B1}.

[2]https://github.com/openvenues/libpostal

[3]The accuracy was computed considering the entire sequence and was not focused on individual tokens.

[4]For a comparison of some of our models with Libpostal, visit our first article [9]

A. Byte-pair Encoding

Byte-pair encoding (BPE) [13] is a data compression algorithm which iteratively replaces the most frequent occurrences of adjacent bytes with a new set of bytes to find a more compact representation of the said data. A new approach for word segmentation based on the BPE algorithm was introduced by [14]. Their technique, which was proposed to solve the OOV problem in Neural Machine Translation (NMT), consists of representing text as a sequence of characters that are iteratively merged using the same reasoning behind BPE. This approach paved the way for the authors to address NMT with an open-vocabulary solution. Another application of BPE is BPEmb [15], a set of embedding models that were trained to produce subword embeddings based on a BPE decomposition of text. BPEmb offers pre-trained models on 275 languages, as well as MultiBPEmb, which is a single model trained on the shared vocabulary of the 275 languages. These models were shown to have a performance similar to other subword embedding techniques on an entity typing task while outperforming these techniques on some languages.

IV. Architectures

The following section describes the architectures of our models, which are composed of an embedding model and a tagging model as shown in Figure 1. We introduce our "base approach" architecture in Subsection IV-A, as well we introduce two improved architectures, one with an attention mechanism (Subsection IV-B) and another using domain adaptation (Subsection IV-C).

A. Base Approach

1) Embedding Model: Since our main objective is to build a single neural network for parsing addresses from multiple countries, it is necessary to have access to embeddings for different languages at runtime. Some libraries, such as fastText [16] and MUSE [17], offer alignment vectors that enable the projection of word embeddings from different languages in the same space. However, these techniques would require detecting the source language as well as specifying the target language to use the proper alignments, which we consider an unnecessary overhead for the task at hand. To resolve the embedding issue, we propose the following two methods.

First, we use a fixed pre-trained monolingual fastText model (pre-trained on the French language). We chose French embeddings since the French language shares Latin roots with many languages in our test set. It is also due to the considerable size of the corpus on which these embeddings were trained. We refer to this embeddings model technique as FastText.

Second, we use an encoding of words using MultiBPEmb and merge the embeddings obtained for each word into a single word embedding using an RNN. This method has been shown to provide good results in a multilingual setting [18]. Our RNN network of choice is a Bidirectional LSTM (Bi-LSTM) with a hidden state dimension of 300. We build the word embeddings by running the concatenated forward and backward hidden states corresponding to the last time step for

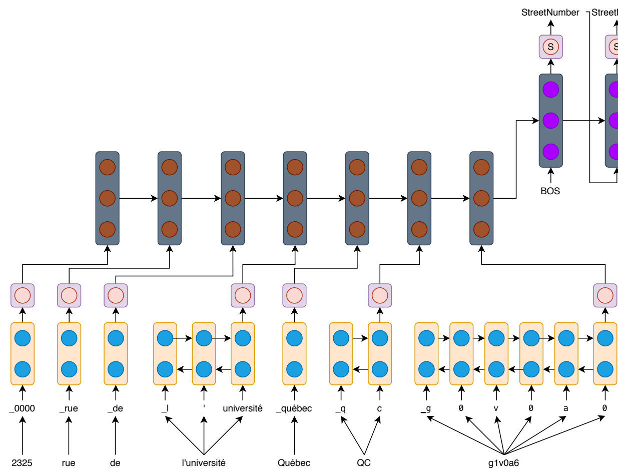

Fig. 1. Illustration of our architecture using the BPEmb embedding model. Each word in the address is encoded using MultiBPEmb (the BPE segmentation algorithm replaces the numbers in the address by zeros). The subword embeddings are fed to the BiLSTM (rounded rectangle with two circles). The last hidden state for each word is run through a fully connected layer (rounded rectangle with one circle). The resulting embeddings are given as input to the Seq2Seq (rounded rectangle with three circles). The 'S' in the fully connected layer following the Seq2Seq decoder stands for the Softmax function.

each word decomposition through a fully connected layer of which the number of neurons is equal to the dimension of the hidden states. This approach produces 300-dimensional word embeddings. We refer to these embeddings as BPEmb.

2) Tagging Model: Our downstream tagging model is a Seq2Seq model consisting of a one-layer unidirectional LSTM encoder and a one-layer unidirectional LSTM decoder followed by a fully-connected linear layer with a softmax activation. Both the encoder's and decoder's hidden states are of dimension 1024. The embedded address sequence is fed to the encoder that produces hidden states, the last of which is used as a context vector to initialize the decoder's hidden states. The decoder is then given a Beginning Of Sequence (BOS) token as input and, at each time step, the prediction from the last step is used as input. To better adapt the model to the task at hand and to facilitate the convergence process, we only require the decoder to produce a sequence with the same length as the input address. This approach differs from the traditional Seq2Seq architecture in which the decoder makes predictions until it predicts the End Of Sequence token. The decoder's outputs are forwarded to the linear layer of which the number of neurons is equal to the tag space dimensionality. The softmax activation function computes probabilities over the linear layer's outputs to predict the most likely token at each time step.

B. Attention Architecture

Attention mechanisms are neural network components that can produce a distribution describing the interdependence between a model's inputs and outputs (general attention) or amongst model inputs themselves (self-attention). These mechanisms are common in natural language processing encoder-decoder architectures such as neural machine translation models [19] since they have shown to improve models' performance and help address some of the issues recurrent neural networks suffer from when it comes to dealing with long sequences.

Attention mechanisms are also exploited for the interpretability of neural networks, where they are considered to provide insights about the impact of some neural network's inputs on its predictions. This use of attention mechanisms has been contested in [20] because of a lack of consistency with feature importance measures, among other things. However, other work has suggested that attention mechanisms provide a certain degree of interpretability depending on the task at hand [21] [22]. In this work, we focus on the performance enhancement of address tagging models using attention.

1) Attention models: The implementation of our attention models was inspired by [19]. The models' architecture remains similar to that of our base models, with some alterations to the decoding process. Indeed, instead of feeding the last predicted tag as an input to the decoder at the current time step i, we compute an input using the encoder's outputs $\vec{O}$, the last decoder hidden state h_{i-1}, and the last predicted tag's representation $\vec{t}$.

We start by computing attention weights as follows:

$$\alpha_{i,j} = \frac{\exp(a_{i,j})}{\sum_k \exp(a_{i,k})}$$

where

$$a_{i,j} = p \times tanh(W_h h_{i-1} + W_o O_j)$$

and W_h, W_o and p are learnable parameters.

Next, we compute a context vector by weighting the encoder's outputs with the obtained attention weights:

$$\vec{c_i} = \sum_k \alpha_{i,k} O_k$$

Finally, we obtain the decoder's input by concatenating the last prediction $\vec{t}$ with the context vector $\vec{c_i}$. Note that the first decoder's input is computed using the last encoder's hidden state.

We use this approach with the two aforementioned embedding methods and name the obtained models **fastTextAtt** and **BPEmbAtt** respectively.

C. Domain Adaptation

Domain adaptation is a branch of transfer learning which aims at applying a model trained on data from a source domain to data from a target domain which somewhat differs from one another but still retains a certain degree of similarity. More specifically, it is a technique used when the input and output belong to the same space, but the probability distribution which associates them changes as we move from one domain to the other [23]. Our objective is to generalize the performance of address parsing models to countries of which no data is used at training time. In order to extend our models to cope with this domain adaptation problem, we enhance our base approach using domain adversarial neural networks.

1) Domain Adversarial Neural Networks: Domain adversarial neural networks [24] achieve domain adaptation by appending a second parallel output layer to a neural network classifier, the purpose of which is to predict the domain of the network's input. Since the target domain labels are not available during training, two losses are computed when the input belongs to the source domain (i.e. the labels classification loss and the domain classification loss), whilst only the domain classification loss is computed when the input belongs to the target domain. Moreover, the gradient associated with the domain classification layer is reversed during backpropagation. This aims to hinder the neural network's ability to differentiate between the source and target domains while still learning to perform well on classifying data from the source domain.

2) Implementation: First of all, we modified our neural network architecture by adding a domain discriminator in the form of a fully connected layer with two output neurons which takes the context vector produced by the Seq2Seq encoder as input. This layer is preceded by a gradient reversal layer that reverses the computed gradient sign during backpropagation. The domain discriminator is followed by a Softmax activation function, and its loss is computed using a Cross-Entropy loss function. Secondly, we used the ADANN [25] training algorithm to train our model since it is designed to enable multi-domain adversarial training by considering, during the forward pass of each batch, one domain as a source domain and another random domain as a target domain, and so on for each of the available domains; our domains being the countries for which training data is available. We hope, by using this approach, to construct models that can learn to parse addresses from countries with different address formats without making a significant distinction between. Therefore these models would perform better on parsing addresses from countries not seen during its training.

We use this approach with the two aforementioned embedding methods and name the obtained models **fastTextADANN** and **BPEmbADANN** respectively.

V. Data

A. Complete Address Dataset

Our dataset was built using the open-source data on which Libpostal's models were trained and of which we have collected the address data of 61 countries. Twenty countries were used for multinational training with a sample size of 100,000 addresses per country, while the rest of the samples was left out as holdout for testing. The other countries' data was also left for zero-shot transfer evaluation. Tables I and II show the number of samples per country in both test sets ordered by number of examples per country. The color in the table will be discussed later on.

We introduce eight tags, namely StreetNumber, StreetName, Unit, Municipality, Province, PostalCode, Orientation, and GeneralDelivery, as opposed to Libpostal, which utilizes 20 tags. This was motivated by the common presence of the chosen tags in most of the countries included in our datasets. Also, it is not guaranteed that all addresses contain each tag category's elements since some addresses might not contain elements of some tag categories. However, all addresses need to have at least each of the following tags: StreetName, PostalCode, Municipality and Province. We will refer to this dataset as the "Complete address" dataset.

Figure 2 shows address samples for different countries with the corresponding tags. Each color represents one of the five different patterns present in our dataset [26]. We also find that some countries' address format is composed of different patterns (e.g. Belarus addresses use the second and fifth patterns). No color is used for these countries.

TABLE I
NUMBER OF SAMPLES PER COUNTRY IN THE HOLDOUT TEST SET FOR COUNTRIES USED FOR TRAINING THE MODELS.

Country	Size	Country	Size	Country	Size	Country	Size
United States	8,000,000	Germany	1,576,059	Poland	459,522	Czechia	195,269
Brazil	8,000,000	Spain	1,395,758	Norway	405,649	Italy	178,848
South Korea	6,048,106	Netherlands	1,202,173	Austria	335,800	France	20,050
Australia	5,428,043	Canada	910,891	Finland	280,219	UK	14,338
Mexico	4,853,349	Switzerland	474,240	Denmark	199,694	Russia	8115

TABLE II
NUMBER OF SAMPLES PER COUNTRY IN THE ZERO-SHOT TEST SET.

Country	Size	Country	Size	Country	Size	Country	Size
Belgium	66,182	Slovenia	9773	Réunion	2514	Bangladesh	888
Sweden	32,291	Ukraine	9554	Moldova	2376	Paraguay	839
Argentina	27,692	Belarus	7590	Indonesia	2259	Bosnia	681
India	26,084	Serbia	6792	Bermuda	2065	Cyprus	836
Romania	19,420	Croatia	5671	Malaysia	2043	Ireland	638
Slovakia	18,975	Greece	4974	South Africa	1388	Algeria	601
Hungary	17,460	New Zealand	4678	Latvia	1325	Colombia	569
Japan	14,089	Portugal	4637	Kazakhstan	1087	Uzbekistan	505
Iceland	13,617	Bulgaria	3716	New Caledonia	1036		
Venezuela	10,696	Lithuania	3126	Estonia	1024		
Philippines	10,471	Faroe Islands	2982	Singapore	968		

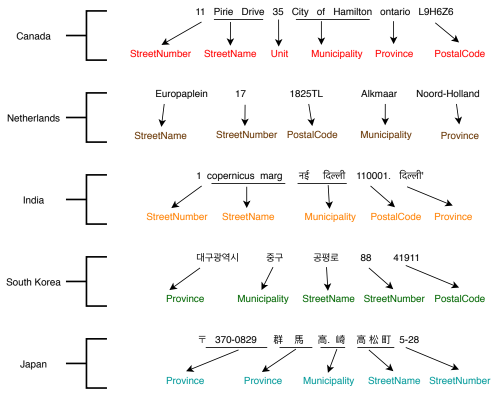

Fig. 2. Examples of the main address patterns found in the dataset.

B. Incomplete Address Dataset

We also introduce a second dataset based on Libpostal's open-source data. It is similar to the Complete address dataset. It is composed of the same twenty countries used for multinational training but with a sample size of 50,000 for training and 25,000 for holdout evaluation. The dataset consists of only addresses where each one is missing at least one of the following four tags: StreetName, PostalCode, Municipality and Province. We consider an address incomplete if it is not composed of at least all of the four tags. For example, the sequence of tags for the address "221 B Baker Street" is {StreetNumber, Unit, StreetName, StreetName}, and it is incomplete since the PostalCode and the Municipality tags are missing. We will refer to this dataset as the "incomplete address" dataset.

VI. Experiments

For our experiments, we trained our six models (**fastText**, **BPEmb**, **fastTextAtt**, **fastTextADANN**, **BPEmbAtt** and **BPEmbADANN**) five times each[5] for 200 epochs with a batch size of 512 for the base approach and the attention models and 256 for the ADANN one. An early stopping with a patience of fifteen epochs was also applied during training. We initialize the learning rate at 0.1 and use learning rate scheduling to lower it by a factor of 0.1 after ten epochs without loss reduction. Our loss function of choice is the Cross-Entropy loss due to its suitability for the softmax function. The optimization is done through Stochastic Gradient Descent.

Also, to speed up the convergence, we use teacher forcing [27], a method that consists of using the ground truth instead of the previous time step's prediction as input for the decoder during training. We do so by randomly sampling part of the training data at runtime. 80 % of the training datasets were used to train the models, and 20 % was kept for validation. The architecture and the training of the models were implemented using Pytorch [28], and Poutyne [29].

A. Evaluation Procedure

We train our six models on our multinational dataset, the difference between the models being (1) the word embedding method employed (fastText and BPEmb) and (2) the use of attention mechanism or domain adaption learning. Each model has been trained five times, and we report the models' mean accuracy and standard deviation on the per-country zero-shot data. The accuracy for each sequence is computed as the proportion of the tags predicted correctly by the model. As such, predicting all the tags for a sequence correctly yields a perfect accuracy. More precisely, errors in tag predictions have an impact on the accuracy for a given sequence. However, the accuracy will not be null unless all the predicted tags for the sequence are incorrect. These results will be discussed in section VII.

B. Incomplete Address Evaluation Procedure

Since addresses do not always include all the components, we will also evaluate four models (**fastTextAtt**, **fastTextADANN**, **BPEmbAtt** and **BPEmbADANN**) on the incomplete addresses dataset introduced in the Subsection V-B. We hypothesize that an incomplete address can confuse our models since we use a seq2seq architecture, and the compressed representation of an incomplete address will not be the same as the same complete one. For example, the address "221 B Baker Street London NW1 6XE" is complete and is a typical way to write an address. But, many addresses are not always in such a form. Such as the address "221 B Baker Street", which is the same as the previous one but without the city and the postal code. That difference can be more challenging for our models since postal code is usually a good way to tell the difference between the pattern shown in Figure 2. We will also evaluate the performance of two new models, **fastTextADANNNoisy** and **BPEmbADANNNoisy**), trained on the complete and incomplete addresses dataset to investigate if the addition of incomplete address help improves performance on that type of data. These results will be discussed in section VIII.

VII. Complete addresses Results

In this section, we present and discuss the results of all our trained models. We first evaluated them on the holdout addresses dataset, and we evaluated them on the zero-shot addresses dataset.

A. Multinational Evaluation

Table III presents all the models' mean accuracy and standard deviation on the holdout dataset for training countries. First, we find that South Korea is the only country where a perfect accuracy was achieved when using byte-pairs embeddings (BPemb) or almost all the time (four seeds out of five) when using fastText embeddings. Since South Korea is the only country using a different pattern in the training set where the province and municipality occur before the street name, it

[5] Using each of the following seeds $\{5, 10, 15, 20, 25\}$. When a model did not converge (a high loss value on train and validation), we retrain the model using a different seed (30).

TABLE III
MEAN ACCURACY AND STANDARD DEVIATION OBTAINED WITH MULTINATIONAL MODELS ON THE HOLDOUT DATASET FOR TRAINING COUNTRIES. **BOLD** VALUES INDICATE THE BEST ACCURACY FOR EACH COUNTRY.

Country	FastText	BPEmb	FastTextAtt	BPEmbAtt	FastTextADANN	BPEmbADANN
United States	99.61 ± 0.09	99.67 ± 0.09	$\mathbf{99.73 \pm 0.02}$	99.65 ± 0.23	99.70 ± 0.03	99.68 ± 0.16
Brazil	99.40 ± 0.10	99.42 ± 0.15	$\mathbf{99.58 \pm 0.04}$	99.42 ± 0.39	99.53 ± 0.04	99.42 ± 0.34
South Korea	99.96 ± 0.01	$\mathbf{100.00 \pm 0.00}$	99.98 ± 0.01	$\mathbf{100.00 \pm 0.00}$	99.98 ± 0.01	$\mathbf{100.00 \pm 0.00}$
Australia	99.68 ± 0.05	$\mathbf{99.80 \pm 0.05}$	99.77 ± 0.03	99.78 ± 0.13	99.76 ± 0.03	99.77 ± 0.17
Mexico	99.60 ± 0.06	99.68 ± 0.06	$\mathbf{99.71 \pm 0.03}$	99.70 ± 0.14	99.68 ± 0.02	99.69 ± 0.12
Germany	99.77 ± 0.04	99.89 ± 0.03	99.85 ± 0.02	99.90 ± 0.08	99.84 ± 0.01	$\mathbf{99.91 \pm 0.05}$
Spain	99.75 ± 0.05	99.85 ± 0.04	99.83 ± 0.02	$\mathbf{99.86 \pm 0.09}$	99.80 ± 0.02	99.83 ± 0.11
Netherlands	99.61 ± 0.07	99.88 ± 0.03	99.75 ± 0.03	99.90 ± 0.07	99.72 ± 0.02	$\mathbf{99.91 \pm 0.05}$
Canada	99.79 ± 0.05	$\mathbf{99.87 \pm 0.04}$	99.87 ± 0.02	99.87 ± 0.10	99.85 ± 0.01	99.86 ± 0.10
Switzerland	99.53 ± 0.09	99.75 ± 0.08	99.62 ± 0.05	99.77 ± 0.14	99.59 ± 0.05	$\mathbf{99.82 \pm 0.12}$
Poland	99.69 ± 0.07	99.89 ± 0.04	99.80 ± 0.02	99.90 ± 0.07	99.78 ± 0.02	$\mathbf{99.92 \pm 0.04}$
Norway	99.46 ± 0.06	98.41 ± 0.63	99.44 ± 0.11	98.20 ± 1.13	$\mathbf{99.53 \pm 0.04}$	97.95 ± 0.44
Austria	99.28 ± 0.03	98.98 ± 0.22	$\mathbf{99.38 \pm 0.06}$	98.96 ± 0.37	99.30 ± 0.07	99.34 ± 0.32
Finland	99.77 ± 0.03	$\mathbf{99.87 \pm 0.01}$	99.83 ± 0.02	99.86 ± 0.01	99.82 ± 0.01	99.84 ± 0.01
Denmark	99.71 ± 0.07	99.90 ± 0.03	99.82 ± 0.03	99.91 ± 0.06	99.80 ± 0.02	$\mathbf{99.93 \pm 0.05}$
Czechia	99.57 ± 0.09	99.89 ± 0.04	99.73 ± 0.02	99.89 ± 0.10	99.70 ± 0.02	$\mathbf{99.90 \pm 0.06}$
Italy	99.73 ± 0.05	99.81 ± 0.05	99.83 ± 0.02	$\mathbf{99.83 \pm 0.11}$	99.80 ± 0.02	99.82 ± 0.11
France	99.66 ± 0.08	99.69 ± 0.11	$\mathbf{99.79 \pm 0.04}$	99.69 ± 0.22	99.77 ± 0.03	99.70 ± 0.17
UK	99.61 ± 0.10	99.74 ± 0.08	$\mathbf{99.77 \pm 0.05}$	99.72 ± 0.20	99.73 ± 0.03	99.72 ± 0.20
Russia	99.03 ± 0.24	$\mathbf{99.67 \pm 0.11}$	99.40 ± 0.13	99.54 ± 0.39	99.23 ± 0.12	99.59 ± 0.31
Mean	99.61 ± 0.20	99.68 ± 0.36	$\mathbf{99.72 \pm 0.16}$	99.67 ± 0.40	99.70 ± 0.18	99.68 ± 0.43

seems that our models might have memorized this particular pattern. To validate this intuition, we randomly reordered 6000 South Korean addresses to follow either the first (red) or the second (brown) address pattern (equally divided between the two). We observe, after this reordering, that the mean accuracy drops to 28.04 % considering that using a random tags annotation, we get a 12.29 % accuracy.

It is also interesting to notice that the models' accuracies are good when using fastText monolingual word embeddings, especially on South Korean addresses despite the entirely different alphabet. These results illustrate that our model, regardless of the embeddings model, learned the representation of an address sequence even if the words' representations are not native to the language (French vs Korean).

Finally, all our models achieve state-of-the-art performance on our holdout dataset while using less data than previous approaches (e.g. Libpostal) and neither pre nor post-processing. However, at this point, it is difficult to conclude which of our models is the leading one. In the following subsection, we investigate the zero-shot performance of our models on countries not seen during training.

B. Zero-Shot Evaluation

Since training a deep learning model to parse addresses from every country in the world would require a significant amount of data and resources, our ongoing work aims at achieving domain adaptation to be able to train on a reasonable amount of data and generalize to data from different sources. We begin by exploring how well our architecture can generalize in a zero-shot manner. To this end, we test each of our FastText and BPEmb trained models on address data from countries not seen during the training. The results are reported in Table IV ordered by dataset size.

TABLE IV
MEAN ACCURACY (AND STANDARD DEVIATION) PER COUNTRY FOR THE ZERO-SHOT TRANSFER MODELS (BASE APPROACH).

Country	FastText	BPEmb	Country	FastText	BPEmb
Belgium	$\mathbf{88.14 \pm 1.04}$	87.29 ± 1.40	Faroe Islands	74.14 ± 1.83	$\mathbf{85.50 \pm 0.11}$
Sweden	81.59 ± 4.53	$\mathbf{90.76 \pm 3.03}$	Réunion	$\mathbf{96.80 \pm 0.45}$	93.67 ± 0.26
Argentina	86.26 ± 0.47	$\mathbf{88.04 \pm 0.83}$	Moldova	$\mathbf{90.18 \pm 0.79}$	86.89 ± 3.01
India	69.09 ± 1.74	$\mathbf{80.04 \pm 3.24}$	Indonesia	64.31 ± 0.84	$\mathbf{70.28 \pm 1.64}$
Romania	$\mathbf{94.49 \pm 1.52}$	91.65 ± 1.21	Bermuda	92.31 ± 0.60	$\mathbf{93.70 \pm 0.35}$
Slovakia	82.10 ± 0.98	$\mathbf{90.31 \pm 3.88}$	Malaysia	78.93 ± 3.78	$\mathbf{94.16 \pm 0.49}$
Hungary	$\mathbf{48.92 \pm 3.59}$	25.51 ± 2.60	South Africa	95.31 ± 1.68	$\mathbf{96.87 \pm 0.96}$
Japan	$\mathbf{41.41 \pm 3.21}$	35.33 ± 1.28	Latvia	$\mathbf{93.66 \pm 0.64}$	74.78 ± 4.33
Iceland	96.55 ± 1.20	$\mathbf{97.38 \pm 1.18}$	Kazakhstan	86.33 ± 3.06	$\mathbf{94.12 \pm 1.94}$
Venezuala	$\mathbf{94.87 \pm 0.53}$	93.05 ± 2.02	New Caledonia	$\mathbf{99.48 \pm 0.15}$	99.25 ± 0.19
Philippines	77.76 ± 3.97	$\mathbf{81.95 \pm 8.07}$	Estonia	$\mathbf{87.08 \pm 1.89}$	77.30 ± 1.22
Slovenia	95.37 ± 0.23	$\mathbf{97.47 \pm 0.45}$	Singapore	86.42 ± 2.36	$\mathbf{86.87 \pm 2.01}$
Ukraine	$\mathbf{92.99 \pm 0.70}$	92.60 ± 1.84	Bangladesh	78.61 ± 0.43	$\mathbf{82.45 \pm 2.54}$
Belarus	91.08 ± 3.08	$\mathbf{96.40 \pm 1.76}$	Paraguay	96.01 ± 1.23	$\mathbf{97.20 \pm 0.35}$
Serbia	$\mathbf{95.31 \pm 0.48}$	92.62 ± 3.83	Cyprus	$\mathbf{97.67 \pm 0.34}$	94.31 ± 7.21
Croatia	$\mathbf{94.59 \pm 2.21}$	88.04 ± 4.68	Bosnia	84.04 ± 1.47	$\mathbf{84.46 \pm 5.76}$
Greece	$\mathbf{81.98 \pm 0.60}$	40.97 ± 14.89	Ireland	$\mathbf{87.44 \pm 0.69}$	86.49 ± 1.31
New Zealand	94.27 ± 1.50	$\mathbf{99.44 \pm 0.29}$	Algeria	$\mathbf{85.37 \pm 2.05}$	84.65 ± 4.47
Portugal	$\mathbf{93.65 \pm 0.46}$	92.68 ± 1.46	Colombia	87.81 ± 0.92	$\mathbf{89.51 \pm 0.88}$
Bulgaria	91.03 ± 2.07	$\mathbf{93.47 \pm 3.07}$	Uzbekistan	$\mathbf{86.76 \pm 1.13}$	75.18 ± 1.92
Lithuania	$\mathbf{87.67 \pm 3.05}$	76.41 ± 1.66			

First, we observe that the **BPEmb** model reaches the highest accuracy most of the time. Indeed, 49 % of the countries tested in zero-shot transfer reached a mean accuracy of at least 90 % using **BPEmb** while using **fastText** only 46 % of the countries reach that same accuracy. Most of these countries share either the same address structure or language proximity with training data. For instance, Venezuela shares the same address pattern as six other countries in the dataset and also shares the same language as Mexico, Spain, and the same Latin root as French. It was also interesting to observe that for Greece, **BPEmb** achieved near half the accuracy result as **fastText**. We hypothesize that for Greece, fastText is able to produce better embeddings from subword units to reach this performance than **BPEmb**.

In contrast, the lowest results (below 70 %) occur for countries where the address pattern and the country's official language were not seen in the training data, such as India, Hungary, and Japan. The last two countries have had the lowest results of all. This is most likely due to the address structure (blue), which is the near inverse of the two most present ones in the training data (red and brown) (Figure 2). Also, those two countries do not share language root with any of the ones present in the training data, making the task difficult for our models. We also see that Kazakhstan, which uses the same address pattern as Japan, achieves better results. The main difference is the official language's (Kazakh and Russian) presence in the training dataset. Moreover, India achieves almost 20 % better results than Hungary and Japan, even if Hindi does not occur in the training dataset. This is probably due to the use of a nearly identical address pattern as the first one (red). The only difference being the inversion of the province and the postal code. It could mean that if no shared language root is present, a shared address structure allows a decent parsing of the address (almost 70 %).

Finally, we observe that **BPEmp** achieves better results than **fastText** for 21 out of 41 countries, where most of them are between 1 % to 20 % better. Considering that nearly 81 % of the countries reach an accuracy above 80 %, we conclude that using BPEmb embeddings gives good results for a zero-shot address parsing task considering that some languages and address patterns do not occur in the training data. These results were surprising to us since **fastText** uses pre-trained embedding trained on a French corpus. We hypothesize that fastText can produce better generalization of embeddings using subword units than **BPEmb**. This highlights that our trained model to combine the **BPEmb** embeddings might have overfitted to our problem due to the dataset size in contrast with FastText embeddings.

1) Attention Models: In an attempt to improve our models' zero-shot performance, the base architecture was augmented with an attention mechanism as described in Subsection IV-B. The results of those two models (**fastTextAtt** and **BPEmbAtt**) are compared to their base approach respectively in Table V. For the model **fastTextAtt** (left section of the table), we observe that attention mechanisms improve the performance for 20 out of 41 countries by more than 1 %. Where 10 out of 41 improves with more than 2 %. Also, 10 out of 41 countries' accuracies were increased by less than 1 %. However, for the other countries (11 out of 41), we observe that the accuracy is less than 0.5 % poorer than the base approach for most of them. The attention mechanism improved two countries' accuracies above the 90 % accuracy (Belgium and Lithuania) and one above the 80 % (Philippines). **FastTextAtt** achieves good results for 83 % of the countries (34), almost 52 % of which (21) are near state-of-the-art performance.

In contrast, the results for the **BPEmbAtt** model (right section of the table) are not as good. We observe that results were be increased by at least more than 1 % for 14 out of 41 countries. While 11 out of 41 saws improve by more than 2 %. Also, 9 out of 41 countries improved by less than 1 %. However, for the other countries (20), we observe that the accuracy is less than 0.5 % poorer than the base approach for half of them. For the other half, results are between 1 % and 2 % lower, thus lowering performance for two countries (India and Bangladesh) below 80 % and one (Malaysia) below 90 %. Thus, the use of attention mechanism using byte-pair multilingual embeddings lowers performance overall, especially since only one country (Estonia) yields results above the 80 % compare to the base approach.
bpembatt achieve good results for 78 % of the countries (32), almost 50 % of which (20) is near state-of-the-art performance.

Finally, we observe that in some cases, the use of attention mechanism can substantially improve performance, such as Greece where the increase is 4 % for **fastTextAtt** and near 16 % for **BPEmbAtt**. We also observe a smaller variance for both the model using the attention mechanisms, meaning those models are more stable during training and converge to a better optimum. Overall, these results show that our attention mechanisms architecture can generally yield better accuracies during training.

2) ADANN: In a second attempt to improve our models' zero-shot performance, the base architecture was augmented with a domain adaptation approach as described in Subsection IV-C. The results of those two models (**fastTextADANN** and **BPEmbADANN**) are compared to their attention approach equivalents in Table V. For both the **fastTextADANN** and **BPEmbADANN** models (left and right section of the table respectively), we observe similar results.

First, the ADANN algorithm can improve the performance for a minority (4 and 8 respectively for **fastTextADANN** and **BPEmbADANN** respectively) of the countries by more than 1 %. A few (2 or 3) out of the 41 improve by more than 2 % for both models. Also, near a fourth of the 41 countries' accuracies increased by less than 1 %. For the other countries (the majority), we observe that for half of them, the difference is less than 1 % poorer than the attention approaches, and the other half is a couple of percent poorer. Sometimes the difference can be as much as 5 % (e.g. Sweden for **fastTextADANN** or Estonia for **BPEmbADANN**). Also, for both the models, two countries drop below 80 % (Lithuania and Moldova for **fastTextADANN** and Philippines and Bosnia for **BPEmbADANN**) and for **BPEmbADANN** Sweden accuracy's drop below 90 %. Thus, the use of domain adaptation technique during training lowers the performance overall, especially since only one country for each model (Malaysia and India respectively) yields results above 80 % compared to the base approach and one above 90 % (Kazakhstan and Malaysia respectively). Although, overall, the two models achieve good results for nearly 80 % of the countries, almost 50 % of which are near state-of-the-art performance. However, these results are less good than models using attention mechanisms in many cases.

Second, we observe that our model does not seem to have fit the training data as much as possible, as shown in Table VII. This table presents the multinational models' mean accuracy (and a standard deviation) on the holdout dataset for **fastTextADANN**. Nevertheless, we are surprised by our results since the ADANN algorithm is a transfer learning

TABLE V
MEAN ACCURACY (AND STANDARD DEVIATION) PER COUNTRY FOR ZERO-SHOT TRANSFER MODELS - BASE APPROACH VERSUS ATTENTION MODELS. **BOLD** VALUES INDICATE THE BEST ACCURACY FOR EACH EMBEDDINGS TYPE.

Country	FastText	FastTextAtt	BPEmb	BPEmbAtt	Country	FastText	FastTextAtt	BPEmb	BPEmbAtt
Belgium	88.14 ± 1.04	$\mathbf{90.01 \pm 1.59}$	87.29 ± 1.40	$\mathbf{89.35 \pm 0.48}$	Faroe Islands	$\mathbf{74.14 \pm 1.83}$	74.00 ± 1.00	85.50 ± 0.11	$\mathbf{85.89 \pm 1.08}$
Sweden	81.59 ± 4.53	$\mathbf{85.73 \pm 2.30}$	$\mathbf{90.76 \pm 3.03}$	90.64 ± 5.07	Réunion	96.80 ± 0.45	$\mathbf{96.86 \pm 1.27}$	93.67 ± 0.26	$\mathbf{94.19 \pm 0.27}$
Argentina	86.26 ± 0.47	$\mathbf{87.76 \pm 0.71}$	$\mathbf{88.04 \pm 0.83}$	87.45 ± 2.57	Moldova	90.18 ± 0.79	$\mathbf{91.25 \pm 0.84}$	86.89 ± 3.01	$\mathbf{89.99 \pm 2.22}$
India	69.09 ± 1.74	$\mathbf{71.94 \pm 2.06}$	$\mathbf{80.04 \pm 3.24}$	79.73 ± 3.65	Indonesia	64.31 ± 0.84	$\mathbf{65.98 \pm 0.97}$	70.28 ± 1.64	$\mathbf{72.05 \pm 2.81}$
Romania	94.49 ± 1.52	$\mathbf{96.01 \pm 0.73}$	91.65 ± 1.21	$\mathbf{93.01 \pm 0.84}$	Bermuda	92.31 ± 0.60	$\mathbf{92.82 \pm 0.68}$	93.70 ± 0.35	$\mathbf{93.82 \pm 0.08}$
Slovakia	82.10 ± 0.98	$\mathbf{84.16 \pm 2.60}$	90.31 ± 3.88	$\mathbf{92.60 \pm 4.64}$	Malaysia	$\mathbf{78.93 \pm 3.78}$	75.79 ± 2.77	$\mathbf{94.16 \pm 0.49}$	89.06 ± 5.22
Hungary	$\mathbf{48.92 \pm 3.59}$	48.48 ± 4.55	$\mathbf{25.51 \pm 2.60}$	24.00 ± 1.09	South Africa	$\mathbf{95.31 \pm 1.68}$	94.82 ± 2.79	$\mathbf{96.87 \pm 0.96}$	96.83 ± 2.11
Japan	41.41 ± 3.21	$\mathbf{42.63 \pm 4.13}$	35.33 ± 1.28	$\mathbf{42.81 \pm 5.73}$	Latvia	$\mathbf{93.66 \pm 0.64}$	93.04 ± 0.85	$\mathbf{74.78 \pm 4.33}$	72.14 ± 2.00
Iceland	$\mathbf{96.55 \pm 1.20}$	96.51 ± 0.50	$\mathbf{97.38 \pm 1.18}$	96.77 ± 0.43	Kazakhstan	86.33 ± 3.06	$\mathbf{89.52 \pm 4.82}$	$\mathbf{94.12 \pm 1.94}$	92.60 ± 5.40
Venezuela	94.87 ± 0.53	$\mathbf{95.37 \pm 0.45}$	93.05 ± 2.02	$\mathbf{93.79 \pm 3.07}$	New Caledonia	99.48 ± 0.15	$\mathbf{99.52 \pm 0.08}$	99.25 ± 0.19	$\mathbf{99.38 \pm 0.34}$
Philippines	77.76 ± 3.97	$\mathbf{82.87 \pm 5.17}$	$\mathbf{81.95 \pm 8.07}$	81.30 ± 3.50	Estonia	87.08 ± 1.89	$\mathbf{89.29 \pm 2.49}$	77.30 ± 1.22	$\mathbf{80.24 \pm 5.49}$
Slovenia	95.37 ± 0.23	$\mathbf{95.74 \pm 0.44}$	$\mathbf{97.47 \pm 0.45}$	97.04 ± 0.38	Singapore	$\mathbf{86.42 \pm 2.36}$	86.40 ± 2.59	86.87 ± 2.01	$\mathbf{87.53 \pm 1.30}$
Ukraine	92.99 ± 0.70	$\mathbf{93.45 \pm 0.76}$	92.60 ± 1.84	$\mathbf{93.52 \pm 1.81}$	Bangladesh	$\mathbf{78.61 \pm 0.43}$	76.48 ± 1.06	$\mathbf{82.45 \pm 2.54}$	79.01 ± 2.18
Belarus	91.08 ± 3.08	$\mathbf{94.43 \pm 3.59}$	$\mathbf{96.40 \pm 1.76}$	94.44 ± 4.19	Paraguay	96.01 ± 1.23	$\mathbf{96.68 \pm 0.32}$	$\mathbf{97.20 \pm 0.35}$	96.25 ± 1.17
Serbia	95.31 ± 0.48	$\mathbf{95.68 \pm 0.33}$	92.62 ± 3.83	$\mathbf{93.09 \pm 1.81}$	Cyprus	$\mathbf{97.67 \pm 0.34}$	97.57 ± 0.31	94.31 ± 7.21	$\mathbf{98.32 \pm 0.37}$
Croatia	94.59 ± 2.21	$\mathbf{96.40 \pm 0.57}$	88.04 ± 4.68	$\mathbf{90.66 \pm 3.76}$	Bosnia	84.04 ± 1.47	$\mathbf{87.42 \pm 1.95}$	84.46 ± 5.76	$\mathbf{88.61 \pm 5.06}$
Greece	81.98 ± 0.60	$\mathbf{85.00 \pm 1.61}$	40.97 ± 14.89	$\mathbf{56.01 \pm 10.98}$	Ireland	87.44 ± 0.69	$\mathbf{87.69 \pm 0.95}$	86.49 ± 1.31	$\mathbf{87.56 \pm 3.01}$
New Zealand	94.27 ± 1.50	$\mathbf{95.91 \pm 1.41}$	$\mathbf{99.44 \pm 0.29}$	98.21 ± 1.37	Algeria	85.37 ± 2.05	$\mathbf{86.03 \pm 1.55}$	84.65 ± 4.47	$\mathbf{85.08 \pm 2.50}$
Portugal	93.65 ± 0.46	$\mathbf{94.54 \pm 0.67}$	92.68 ± 1.46	$\mathbf{93.33 \pm 0.59}$	Colombia	87.81 ± 0.92	$\mathbf{89.09 \pm 0.64}$	$\mathbf{89.51 \pm 0.88}$	88.32 ± 2.55
Bulgaria	$\mathbf{91.03 \pm 2.07}$	90.87 ± 2.63	$\mathbf{93.47 \pm 3.07}$	92.97 ± 3.66	Uzbekistan	86.76 ± 1.13	$\mathbf{87.36 \pm 0.74}$	75.18 ± 1.92	$\mathbf{77.52 \pm 2.62}$
Lithuania	87.67 ± 3.05	$\mathbf{90.88 \pm 1.73}$	$\mathbf{76.41 \pm 1.66}$	76.16 ± 1.54					

technique. An advantage of ADANN is that the network's weights have strong incentives to be subject-agnostic, meaning that the learned representation extracted from the network can be thought of as general features for the prediction layer [25]. We argue that it is more challenging to train models using the ADANN algorithm since the time needed for one epoch is nearly 5 hours. Meaning that the expected time to train for the 200 epochs is near 41 days per model (and we train five seeds) compared to a couple of days for the attention models. So we did not have much opportunity to fine-tune our model. We also hypothesize that the domain choice (i.e. the country) might be too granular since many countries have similar patterns or similar language, making the task more difficult. An idea of improvement could be to use more definitions for the domain, such as the language and the address pattern type. For example, we could use a categorical variable representing the address pattern number and a categorical variable for the language. That way, we could help guide the training into a better understanding of the context of an address which is not necessary the country but rather the language and the pattern.

Finally, on average, performance is similar between the attention and ADANN approaches, but FastText models perform slightly more. They are, on average, 2 % better than the BPEmb approaches. For example, **fastTextADANN** yields in average 86.45 ± 12.13 % accuracy across the zero-shot countries and **BPEmbADANN** yields a 85.32 ± 15.31 %. Again, both BPEmb models have higher variance than FastText one. These results show that BPEmb models' performance tends to be more variable and more sensitive to changes in seeds.

VIII. MISSING VALUES HANDLING

In this section, we aim to evaluate and improve the results of our four best models, **fastTextAtt**, **BPEmbAtt**, **fastTextADANN** and **BPEmbADANN** on the incomplete address data. As presented in Subsection V-B, we introduce an incomplete address dataset where the addresses do not always include all the components. Table VIII presents the results of the four models evaluated on the incomplete holdout dataset without any prior training on incomplete addresses. Since we observe that performances for all of the training countries are lower by 20 % to 40 % than previous scores, we choose to only evaluate our models on the countries seen during training (holdout). The lower accuracy is South Korea for both the models (nearly over the 45 % accuracy)(will be discussed in more detail later). We also observe that the ADANN approach yields better results (12 out of 20) and is better by less than 1 % on average (last row). Finally, we observe that the BPEmb approach still has higher variance than the FastText one since we have observed some seeds converging to suboptimal loss. Nevertheless, we observe that despite poorer performance on zero-shot evaluation, the ADANN approaches yield better results on incomplete addresses than attention approaches. We hypothesize that ADANN models have not overfitted the representation of an address structure seen in the training and more on the general features of address structures and domain type (e.g. the country and language). Also, knowing the domain, and indirectly the address structure and language, makes it easier to parse an address when incomplete.

To improve our models' incomplete addresses performance, we have trained two new models using the complete and incomplete addresses datasets. This merged dataset consists of 150,000 addresses per country using the same settings as described in section VI. We choose to only train the two best models on incomplete addresses, **fastTextADANN** and **BPEmbADANN**. We refer to these new models by **fastTextADANNNoisy** and **BPEmbADANNNoisy** where the difference between the two is the embeddings. Table IX present the mean accuracy and a standard deviation of our two models tested on the incomplete dataset.

First, we observe that using incomplete addresses during training substantially improves the accuracy for all the coun-

TABLE VI
MEAN ACCURACY (AND STANDARD DEVIATION) PER COUNTRY FOR ZERO-SHOT TRANSFER MODELS - ATTENTION MODELS VERSUS ADANN MODELS.

Country	FastTextAtt	FastTextADANN	BPEmbAtt	BPEmbADANN	Country	FastTextAtt	FastTextADANN	BPEmbAtt	BPEmbADANN
Belgium	90.01 ± 1.59	**90.90 ± 3.76**	**89.35 ± 0.48**	88.74 ± 0.59	Faroe Islands	**74.00 ± 1.00**	69.76 ± 2.08	**85.89 ± 1.08**	85.34 ± 0.20
Sweden	**85.73 ± 2.30**	80.99 ± 3.78	**90.64 ± 5.07**	89.65 ± 1.08	Réunion	**96.86 ± 1.27**	96.48 ± 0.96	**94.19 ± 0.27**	93.99 ± 0.59
Argentina	**87.76 ± 0.71**	87.72 ± 1.33	**87.45 ± 2.57**	87.13 ± 2.65	Moldova	**91.25 ± 0.84**	89.87 ± 0.56	**89.99 ± 2.22**	88.67 ± 1.66
India	**71.94 ± 2.06**	70.30 ± 1.22	79.73 ± 3.65	**81.12 ± 2.97**	Indonesia	**65.98 ± 0.97**	65.42 ± 0.75	**72.05 ± 2.81**	71.71 ± 2.79
Romania	96.01 ± 0.73	**96.22 ± 1.11**	**93.01 ± 0.84**	92.58 ± 1.53	Bermuda	**92.82 ± 0.68**	92.02 ± 0.52	93.82 ± 0.08	**95.02 ± 0.58**
Slovakia	**84.16 ± 2.60**	81.93 ± 1.80	**92.60 ± 4.64**	91.14 ± 3.81	Malaysia	75.79 ± 2.77	**83.48 ± 3.42**	89.06 ± 5.22	**90.52 ± 4.27**
Hungary	**48.48 ± 4.55**	46.42 ± 5.62	**24.00 ± 1.09**	23.54 ± 1.69	South Africa	**94.82 ± 2.79**	93.09 ± 2.35	**96.83 ± 2.11**	96.02 ± 1.55
Japan	42.63 ± 4.13	**43.88 ± 1.26**	**42.81 ± 5.73**	38.64 ± 5.26	Latvia	**93.04 ± 0.85**	90.64 ± 1.26	**72.14 ± 2.00**	71.92 ± 2.60
Iceland	**96.51 ± 0.50**	96.18 ± 0.35	96.77 ± 0.43	**97.15 ± 0.64**	Kazakhstan	89.52 ± 4.82	**92.00 ± 1.36**	92.60 ± 5.40	**96.49 ± 1.94**
Venezuela	**95.37 ± 0.45**	95.02 ± 0.76	93.79 ± 3.07	**94.79 ± 0.57**	New Caledonia	**99.52 ± 0.08**	99.52 ± 0.11	99.38 ± 0.34	**99.40 ± 0.26**
Philippines	82.87 ± 5.17	**83.81 ± 4.95**	**81.30 ± 3.50**	74.94 ± 5.57	Estonia	**89.29 ± 2.49**	86.72 ± 4.00	80.24 ± 5.49	**85.84 ± 4.87**
Slovenia	95.74 ± 0.44	**95.75 ± 0.53**	**97.04 ± 0.38**	96.77 ± 0.57	Singapore	**86.40 ± 2.59**	84.96 ± 2.84	**87.53 ± 1.30**	84.70 ± 1.86
Ukraine	93.45 ± 0.76	**93.70 ± 0.28**	93.52 ± 1.81	**94.17 ± 2.30**	Bangladesh	76.48 ± 1.06	**76.80 ± 1.21**	**79.01 ± 2.18**	78.68 ± 3.96
Belarus	94.43 ± 3.59	**96.05 ± 0.57**	94.44 ± 4.19	**98.15 ± 0.97**	Paraguay	96.68 ± 0.32	**97.00 ± 1.08**	96.25 ± 1.17	**96.28 ± 0.96**
Serbia	95.68 ± 0.33	**95.75 ± 0.25**	93.09 ± 1.81	**93.45 ± 1.44**	Cyprus	**97.57 ± 0.31**	97.39 ± 0.38	**98.32 ± 0.37**	97.53 ± 1.49
Croatia	**96.40 ± 0.57**	94.98 ± 2.01	90.66 ± 3.76	**91.37 ± 4.39**	Bosnia	**87.42 ± 1.95**	84.95 ± 3.47	**88.61 ± 5.06**	79.11 ± 3.07
Greece	**85.00 ± 1.61**	83.18 ± 3.31	56.01 ± 10.98	**57.47 ± 7.75**	Ireland	**87.69 ± 0.95**	87.44 ± 0.20	87.56 ± 3.01	**88.42 ± 0.97**
New Zealand	**95.91 ± 1.41**	93.60 ± 2.42	98.21 ± 1.37	**99.13 ± 0.37**	Algeria	**86.03 ± 1.55**	83.07 ± 3.85	**85.08 ± 2.50**	83.64 ± 2.60
Portugal	**94.54 ± 0.67**	94.53 ± 0.41	**93.33 ± 0.59**	91.07 ± 1.26	Colombia	**89.09 ± 0.64**	87.76 ± 1.30	88.32 ± 2.55	**88.99 ± 1.83**
Bulgaria	**90.87 ± 2.63**	90.40 ± 1.10	92.97 ± 3.66	**93.93 ± 2.59**	Uzbekistan	**87.36 ± 0.74**	86.23 ± 2.18	**77.52 ± 2.62**	73.83 ± 2.92
Lithuania	**90.88 ± 1.73**	88.58 ± 2.35	76.16 ± 1.54	**77.26 ± 2.54**					

TABLE VII
MEAN ACCURACY (AND STANDARD DEVIATION) FOR MULTINATIONAL MODELS ON HOLDOUT DATASET FOR TRAINING COUNTRIES - **FASTTEXTATT** VERSUS **FASTTEXTADANN**.

Country	FastTextAtt	FastTextADANN	Country	FastTextAtt	FastTextADANN
United States	**99.73 ± 0.02**	99.70 ± 0.03	Poland	**99.80 ± 0.02**	99.78 ± 0.02
Brazil	**99.58 ± 0.04**	99.53 ± 0.04	Norway	99.44 ± 0.11	**99.53 ± 0.04**
South Korea	**99.98 ± 0.01**	**99.98 ± 0.01**	Austria	**99.38 ± 0.06**	99.30 ± 0.07
Australia	**99.77 ± 0.03**	99.76 ± 0.03	Finland	**99.83 ± 0.02**	99.82 ± 0.01
Mexico	**99.71 ± 0.03**	99.68 ± 0.02	Denmark	**99.82 ± 0.03**	99.80 ± 0.02
Germany	**99.85 ± 0.02**	99.84 ± 0.01	Czechia	**99.73 ± 0.02**	99.70 ± 0.02
Spain	**99.83 ± 0.02**	99.80 ± 0.02	Italy	**99.83 ± 0.02**	99.80 ± 0.02
Netherlands	**99.75 ± 0.03**	99.72 ± 0.02	France	**99.79 ± 0.04**	99.77 ± 0.03
Canada	**99.87 ± 0.02**	99.85 ± 0.01	UK	**99.77 ± 0.05**	99.73 ± 0.03
Switzerland	**99.62 ± 0.05**	99.59 ± 0.05	Russia	**99.40 ± 0.13**	99.23 ± 0.12

tries. We also observe that the **fastTextADANNNoisy** is the leading model across the board with the best accuracy on all the twenty countries, where results are nearly always more than 98 % in accuracy. In contrast, we observe that, again, despite using an embeddings layer to learn the representation of the byte-pairs embeddings (BPEmb), **BPEmbADANNNoisy** shows poor results compare to **fastTextADANNNoisy**. Results are, on average, nearly 6 % worse and have a higher variance of more than the double, with results as low as 84 % (which is lower than some results observe in zero-shot evaluation). Again, this shows that the trained embeddings layer is overfitting or that the byte-pair embeddings are not well suited for address (such as the embeddings of postal code). This also could mean that the French fastText embeddings approach to construct out-of-vocabulary embeddings are more generalized than the one that we retrain using multi-lingual embeddings.

Second, it is interesting to note that even with a relatively large number of incomplete addresses in the training dataset (50,000), we did not achieve scores as good as seen with the complete dataset (Table III). Results are in average of 98 % an near 93 % for **fastTextADANN** and **BPEmbADANN** respectively. Also, accuracies on the complete addresses holdout dataset are nearly as good as those presented in Table III. We observe similar results for most of the training countries with a difference of 0.5 % lower.

Finally, we can see that the worse results for both models are for South Korea. These results contrast with the nearly perfect score observed for all the models in Table III. This highlights our hypothesis (Subsection VII-A) that our model has memorized the particular pattern of South Korea during training for complete address. However, since we also have trained using incomplete addresses, some incomplete addresses are now not so different from the other address patterns, confusing our models. For example, if we remove the tags Province and Municipality of an address, it can be in 4 of the five patterns making more harder for our models. This shows that our models have overfitted for that case, and adding noise in the training data helps reduce that overfitting but lowers the accuracy. We also observe that using a domain adversarial technique substantially improves performance for that specific case where we observe the best improvement with the accuracy nearly doubling.

IX. CONCLUSION

We estimate that we have reached our first objective, which was to build a single model capable of learning to parse addresses of different formats and languages using a multinational dataset and subword embeddings. Indeed, all our approaches achieve accuracies around 99 % on all the twenty countries used for training. Our experiments with zero-shot transfer learning also yielded interesting results. First, our baseline approaches obtain good results achieving near 50 % of state-of-the-art performance. Second, using an attention mechanism helps to improve our results and could also provide insights about the address elements on which the model focuses to make a tag prediction. However, this analysis is left as future work. Third, our experiments indicate that using a domain adversarial training algorithm does not necessarily improve our results on countries not seen during training. However, they bring a significant contribution on incomplete addresses. Finally, we tested some of our models on incomplete addresses to evaluate their performance on

TABLE VIII
MEAN ACCURACY (AND STANDARD DEVIATION) FOR MULTINATIONAL MODELS WITHOUT ANY PRIOR TRAINING ON INCOMPLETE DATASET.

Country	FastTextAtt	FastTextADANN	BPEmbAtt	BPEmbADANN
United States	$\mathbf{68.59 \pm 2.61}$	65.57 ± 0.80	66.14 ± 3.88	$\mathbf{67.94 \pm 4.94}$
Brazil	57.00 ± 2.41	$\mathbf{57.39 \pm 3.63}$	$\mathbf{55.01 \pm 3.90}$	51.10 ± 4.08
South Korea	45.91 ± 4.22	$\mathbf{46.28 \pm 2.29}$	$\mathbf{49.84 \pm 8.17}$	35.85 ± 5.32
Australia	75.90 ± 1.28	$\mathbf{76.02 \pm 0.36}$	$\mathbf{77.18 \pm 0.80}$	76.49 ± 1.56
Mexico	$\mathbf{61.52 \pm 2.42}$	59.88 ± 2.83	$\mathbf{61.25 \pm 1.42}$	58.93 ± 6.36
Germany	48.39 ± 1.40	$\mathbf{50.32 \pm 6.96}$	$\mathbf{58.98 \pm 0.86}$	58.62 ± 3.31
Spain	70.74 ± 1.26	$\mathbf{72.39 \pm 1.13}$	79.51 ± 0.49	$\mathbf{81.05 \pm 1.17}$
Netherlands	59.72 ± 7.54	$\mathbf{63.74 \pm 3.47}$	75.34 ± 1.32	$\mathbf{77.39 \pm 3.52}$
Canada	$\mathbf{72.77 \pm 0.76}$	71.04 ± 1.98	73.00 ± 2.51	$\mathbf{81.29 \pm 3.22}$
Switzerland	$\mathbf{64.13 \pm 6.21}$	62.66 ± 6.31	$\mathbf{73.08 \pm 0.89}$	72.55 ± 3.22
Poland	43.99 ± 3.35	$\mathbf{47.04 \pm 4.06}$	48.50 ± 1.08	$\mathbf{54.47 \pm 4.73}$
Norway	$\mathbf{64.69 \pm 9.77}$	62.67 ± 8.64	76.43 ± 1.62	$\mathbf{78.28 \pm 4.04}$
Austria	$\mathbf{69.33 \pm 6.67}$	69.21 ± 5.21	77.69 ± 1.08	$\mathbf{78.56 \pm 2.39}$
Finland	57.94 ± 8.91	$\mathbf{60.83 \pm 6.98}$	74.65 ± 1.36	$\mathbf{76.31 \pm 3.46}$
Denmark	$\mathbf{56.94 \pm 3.24}$	55.54 ± 5.48	$\mathbf{65.32 \pm 2.40}$	63.73 ± 3.04
Czechia	61.16 ± 3.21	$\mathbf{64.40 \pm 2.91}$	72.58 ± 1.72	$\mathbf{74.79 \pm 3.53}$
Italy	66.42 ± 2.49	$\mathbf{71.16 \pm 1.67}$	73.89 ± 0.77	$\mathbf{76.33 \pm 1.46}$
France	70.15 ± 0.88	$\mathbf{73.14 \pm 2.38}$	71.91 ± 3.46	$\mathbf{77.45 \pm 1.08}$
UK	53.63 ± 1.28	$\mathbf{54.58 \pm 1.24}$	51.27 ± 1.96	$\mathbf{53.62 \pm 3.30}$
Russia	$\mathbf{58.41 \pm 1.12}$	57.94 ± 0.99	58.80 ± 5.21	$\mathbf{61.91 \pm 2.86}$
Mean	61.37 ± 8.65	$\mathbf{62.09 \pm 8.41}$	67.02 ± 9.91	$\mathbf{67.83 \pm 12.15}$

TABLE IX
MEAN ACCURACY (AND STANDARD DEVIATION) FOR MULTINATIONAL MODELS TRAINED ON COMPLETE AND INCOMPLETE DATASETS AND EVALUATED ON THE INCOMPLETE DATASET FOR TRAINING COUNTRIES. **BOLD** VALUES INDICATE THE BEST ACCURACY FOR EACH COUNTRY.

Country	FastTextADANN	FastTextADANNNoisy	BPEmbADANN	BPEmbADANNNoisy
United States	65.57 ± 0.80	$\mathbf{97.88 \pm 2.04}$	67.94 ± 4.94	92.12 ± 3.60
Brazil	57.39 ± 3.63	$\mathbf{98.39 \pm 1.90}$	51.10 ± 4.08	89.57 ± 6.46
South Korea	46.28 ± 2.29	$\mathbf{92.25 \pm 3.72}$	35.85 ± 5.32	84.58 ± 9.86
Australia	76.02 ± 0.36	$\mathbf{98.86 \pm 1.46}$	76.49 ± 1.56	93.32 ± 5.75
Mexico	59.88 ± 2.83	$\mathbf{97.56 \pm 1.63}$	58.93 ± 6.36	88.79 ± 5.12
Germany	50.32 ± 6.96	$\mathbf{99.20 \pm 0.74}$	58.62 ± 3.31	96.79 ± 2.05
Spain	72.39 ± 1.13	$\mathbf{98.24 \pm 1.82}$	81.05 ± 1.17	90.23 ± 7.02
Netherlands	63.74 ± 3.47	$\mathbf{98.65 \pm 0.98}$	77.39 ± 3.52	97.21 ± 1.77
Canada	71.04 ± 1.98	$\mathbf{97.88 \pm 2.65}$	81.29 ± 3.22	91.93 ± 3.74
Switzerland	62.66 ± 6.31	$\mathbf{99.03 \pm 0.78}$	72.55 ± 3.22	97.12 ± 1.52
Poland	47.04 ± 4.06	$\mathbf{99.28 \pm 0.52}$	54.47 ± 4.73	97.89 ± 1.80
Norway	62.67 ± 8.64	$\mathbf{99.35 \pm 0.54}$	78.28 ± 4.04	98.25 ± 1.06
Austria	69.21 ± 5.21	$\mathbf{99.10 \pm 1.16}$	78.56 ± 2.39	94.74 ± 2.88
Finland	60.83 ± 6.98	$\mathbf{98.97 \pm 0.64}$	76.31 ± 3.46	98.75 ± 0.57
Denmark	55.54 ± 5.48	$\mathbf{97.38 \pm 2.56}$	63.73 ± 3.04	92.15 ± 3.77
Czechia	64.40 ± 2.91	$\mathbf{98.14 \pm 1.74}$	74.79 ± 3.53	93.75 ± 3.79
Italy	71.16 ± 1.67	$\mathbf{98.82 \pm 1.15}$	76.33 ± 1.46	94.49 ± 3.72
France	73.14 ± 2.38	$\mathbf{98.98 \pm 1.32}$	77.45 ± 1.08	91.10 ± 6.85
UK	54.58 ± 1.24	$\mathbf{96.01 \pm 4.75}$	53.62 ± 3.30	84.95 ± 7.63
Russia	57.94 ± 0.99	$\mathbf{96.05 \pm 4.05}$	61.91 ± 2.86	86.71 ± 5.71
Mean	62.09 ± 8.41	$\mathbf{98.00 \pm 1.62}$	67.83 ± 12.15	92.72 ± 4.22

such data. We also improved performance by using some incomplete addresses during training improved performance. These results provide insights into the direction that our future work should take. It would be interesting to explore how other subword embeddings techniques, such as character-based ones [30], would perform on the multinational address parsing task. Additional qualitative analysis of the results would also be required to investigate the models' typical errors further.

ACKNOWLEDGMENT

This research was supported by the Natural Sciences and Engineering Research Council of Canada (IRCPJ 529529-17) and a Canadian insurance company. We wish to thank the reviewers for their comments regarding our work and methodology.

REFERENCES

[1] S. Xu, S. Flexner, and V. R. Carvalho, "Geocoding billions of addresses: Toward a spatial record linkage system with big data," 2012.
[2] N. Abid, A. ul Hasan, and F. Shafait, "DeepParse: A Trainable Postal Address Parser," in *Digital Image Computing: Techniques and Applications)*, 2018, pp. 1–8.
[3] S. Mokhtari, A. Mahmoody, D. Yankov, and N. Xie, "Tagging Address Queries in Maps Search," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 9547–9551.
[4] X. Li, H. Kardes, X. Wang, and A. Sun, "HMM-Based Address Parsing: Efficiently Parsing Billions of Addresses on MapReduce," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference*

on Advances in Geographic Information Systems. Association for Computing Machinery, 2014, p. 433–436.

[5] M. Wang, V. Haberland, A. Yeo, A. Martin, J. Howroyd, and J. M. Bishop, "A Probabilistic Address Parser Using Conditional Random Fields and Stochastic Regular Grammar," in *16th International Conference on Data Mining Workshops*, 2016, pp. 225–232.

[6] S. Sharma, R. Ratti, I. Arora, A. Solanki, and G. Bhatt, "Automated Parsing of Geographical Addresses: A Multilayer Feedforward Neural Network Based Approach," in *IEEE 12th International Conference on Semantic Computing*, 2018, pp. 123–130.

[7] A. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, p. 260–269, Sep. 1967.

[8] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[9] M. Yassine, D. Beauchemin, F. Laviolette, and L. Lamontagne, "Leveraging subword embeddings for multinational address parsing," in *2020 6th IEEE Congress on Information Science and Technology (CiSt)*, 2020, pp. 353–360.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," 2013.

[11] J. Pennington, R. Socher, and C. Manning, "GloVe: Global vectors for word representation," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, Oct. 2014, pp. 1532–1543.

[12] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," 2016.

[13] P. Gage, "A New Algorithm for Data Compression," *C Users Journal*, vol. 12, no. 2, p. 23–38, 1994.

[14] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Aug. 2016, pp. 1715–1725.

[15] B. Heinzerling and M. Strube, "BPEmb: Tokenization-free Pre-trained Subword Embeddings in 275 Languages," 2017.

[16] A. Joulin, P. Bojanowski, T. Mikolov, H. Jégou, and E. Grave, "Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2018.

[17] A. Conneau, G. Lample, M. Ranzato, L. Denoyer, and H. Jégou, "Word Translation Without Parallel Data," 2017.

[18] B. Heinzerling and M. Strube, "Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Jul. 2019, pp. 273–291.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1409.0473

[20] S. Jain and B. C. Wallace, "Attention is not Explanation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 3543–3556. [Online]. Available: https://www.aclweb.org/anthology/N19-1357

[21] S. Wiegreffe and Y. Pinter, "Attention is not not explanation," 2019.

[22] S. Vashishth, S. Upadhyay, G. S. Tomar, and M. Faruqui, "Attention interpretability across nlp tasks," 2019.

[23] W. M. Kouw and M. Loog, "An introduction to domain adaptation and transfer learning," 2019.

[24] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," vol. 17, no. 1, 2016.

[25] U. Côté-Allard, E. Campbell, A. Phinyomark, F. Laviolette, B. Gosselin, and E. Scheme, "Interpreting deep learning features for myoelectric control: A comparison with handcrafted features," *Frontiers in Bioengineering and Biotechnology*, vol. 8, p. 158, 2020. [Online]. Available: https://www.frontiersin.org/article/10.3389/fbioe.2020.00158

[26] G. Rhind, "Global Sourcebook for International Data Management," https://www.grcdi.nl/gsb/global%20sourcebook.html, June 2020.

[27] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[28] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035.

[29] F. Paradis, D. Beauchemin, M. Godbout, M. Alain, N. Garneau, S. Otte, A. Tremblay, M.-A. Bélanger, and F. Laviolette, "Poutyne: A Simplified Framework for Deep Learning," 2020, https://poutyne.org.

[30] Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-Aware Neural Language Models," 2015.