Deepparse : An Extendable, and Fine-Tunable State-Of-The-Art Library for Parsing Multinational Street Addresses

David Beauchemin, Marouane Yassine
Department of Computer Science and Software Engineering, Laval University
Group for Research in Artificial Intelligence of Laval University (GRAIL)
Québec, Canada
david.beauchemin@ift.ulaval.ca, marouane.yassine.1@ulaval.ca

Abstract

Segmenting an address into meaningful components, also known as address parsing, is an essential step in many applications from record linkage to geocoding and package delivery. Consequently, a lot of work has been dedicated to develop accurate address parsing techniques, with machine learning and neural network methods leading the state-of-the-art scoreboard. However, most of the work on address parsing has been confined to academic endeavours with little availability of free and easy-to-use open-source solutions.

This paper presents Deepparse, a Python open-source, extendable, fine-tunable address parsing solution under LGPL-3.0 licence to parse multinational addresses using state-of-the-art deep learning algorithms and evaluated on over 60 countries. It can parse addresses written in any language and use any address standard. The pre-trained model achieves average 99 % parsing accuracies on the countries used for training with no pre-processing nor post-processing needed. Moreover, the library supports fine-tuning with new data to generate a custom address parser.

1 Introduction

Address Parsing is the task of decomposing an address into its different components (Abid et al., 2018). This task is essential to many applications, such as geocoding and record linkage. Indeed, it is quite useful to detect the different parts of an address to find a particular location based on textual data 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 (Xu et al., 2012) to probabilistic approaches and neural network models (Abid et al., 2018), much progress has been made in reaching accurate addresses segmentation. These previous works did a remarkable job of 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 substantial data pre-processing pipelines (Mokhtari et al.; Li et al., 2014; Wang et al., 2016; Sharma et al., 2018).

However, most of the work on address parsing has been confined to academic endeavours with little availability of free and easy-to-use open-source solutions. In an effort to solve some of the limitations of previous methods, as well as offer an open-source address parsing solution, we have created **Deepparse**[1] (Yassine and Beauchemin, 2020) an LGPL-3.0 licenced Python library. Our work allows anyone with a basic knowledge of Python or command line terminal to conveniently parse addresses from multiple countries using state-of-the-art deep learning models proposed by Yassine et al. (2020, 2022). Deepparse's goal is to parse multinational addresses written in any language or using any address writing format with an extendable and fine-tunable address parser. In addition, **Deepparse** proposes a functionality to easily customize the aforementioned models to new data along with an easy-to-use Docker FastAPI to parse addresses.

This paper's contributions are: First, we describe an open-source Python library for multinational address parsing. Second, we describe its implementation details and natural extensibility due to its fine-tuning possibilities. Third, we benchmark it against other open-source libraries.

[1] https://deepparse.org/

2 Related work

Address parsing has been approached on the academic front using probabilistic machine learning models such as Hidden Markov Models and Conditional Random Fields (CRF) (Li et al., 2014; Wang et al., 2016; Abid et al., 2018), as well as deep learning models mainly based on the recurrent neural network (RNN) architecture (Sharma et al., 2018; Mokhtari et al.; Abid et al., 2018). Regarding openly available software, most of the existing packages cater to US postal addresses. For instance, *pyaddress*[2] allows for the decomposition of US addresses into eight different attributes with a possibility to specify acceptable "street names", "cities" and "street suffixes" in order to improve parsing accuracy. Similarly, *address-parser*[3] identifies as "Yet another python address parser for US postal addresses" and enables users to extract multiple address components such as "house numbers", "street names", "cardinal directions" and "zip codes". These two packages are based on a combination of predefined component lists and regular expressions. In contrast, *usaddress*[4] uses a probabilistic model that users can fine-tune using their data. Another openly available avenue for address parsing is Geocoding APIs, which can result in highly precise parsed addresses based on reverse geocoding. However, while being openly available, Geocoding APIs are often not free and not always convenient to use for a programming layperson.

The aforementioned approaches are limited to parsing addresses from a single country and either cannot handle a multinational scope of address parsing or would need to be adjusted to do so. To tackle this problem, Libpostal[5], a C 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 Libpostal dataset[6] and achieved a 99.45 % full parse accuracy[7] using an extensive pre and post-processing pipeline. However, this requires putting addresses through a heavy pre-processing pipeline before feeding them to the prediction model, and it does not seem possible to develop a new address parser based on the documentation. A thorough search of the relevant literature yielded no open-source neural network-based software for multinational address parsing.

3 Implementation

Deepparse is divided into three high-level components: pre-processors, embeddings model, and tagging model. The first component, the pre-processor, is a series of simple handcrafted pre-processing functions to be applied as a data cleaning procedure before the embedding component, such as lowercasing the address text and removing commas. By default, Deepparse simply lowercase and removes all commas in the address. The library does not require a complex pre-processing pipeline, but one can be defined and used more complex one if needed since Deepparse is built so users can handcraft and use a custom pre-processor during this phase.

The last two components are illustrated in Figure 1. We can see that the embeddings model component (black) encodes each token (i.e. word) of the address into a recurrent dense representation. At the end of the sentence, the component generates a single dense representation for the overall address generated from the individual address components. Then, this address-dense representation is used as input to the tagging model component (red), where each address component is decoded and classified into its appropriate tag. These two components do not rely on named entity recognition to parse addresses as opposed to the one proposed by Abid et al. (2018).

Deepparse proposes two embeddings model approaches and four pre-trained tagging model architectures; all approaches can be used with CPU or GPU setup. All pre-trained approaches have been trained on our publicly available dataset[8], based on to the Libpostal dataset, and achieved parse accuracies higher than 99% on the 20 trained countries without using pre or post-processing[9].

The following sub-section will briefly discuss how these two components work. For more details on the algorithms behind both components, readers can refer to Yassine et al. (2020, 2022). We will finish this section with a presentation on Deepparse's unique flexibility in developing a new

[2] https://github.com/SwoopSearch/pyaddress

[3] https://github.com/CivicKnowledge/address_parser

[4] https://github.com/datamade/usaddress

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

[6] https://github.com/openvenues/libpostal#training-data

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

[8] https://github.com/GRAAL-Research/deepparse-address-data

[9] The accuracy for each sequence is computed as the proportion of the tags predicted correctly by the model. Predicting all the tags correctly for a sequence yields perfect accuracy.

address parser.

3.1 Embedding Model

Our objective was to build a single neural network to parse addresses from multiple countries. Thus, access to embeddings for different languages at runtime was necessary. Since the use of alignment vectors (Joulin et al., 2018; Conneau et al., 2017) would have introduced the unnecessary overhead of detecting of the source language to project word embeddings from different languages in the same space, Deepparse proposes the following two methods.

First, we use a fixed pre-trained monolingual French fastText model. We chose French embeddings since this language shares Latin roots with many languages in our test set. It is also due to the large 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 obtained embeddings for each word into one word embedding using an RNN. This method has been shown to give good results in a multilingual setting (Heinzerling and Strube, 2019). 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 each word decomposition through a fully connected layer of which the number of neurons equals the dimension of the hidden states. This approach produces 300-dimensional word embeddings. We refer to this embeddings model technique as **BPEmb**.

3.2 Tagging Model

Our downstream tagging model is a Seq2Seq model. Using Seq2Seq architecture as tagging model is effective for data with sequential pattern (Huang et al., 2019; Omelianchuk et al., 2021; Jin and Yu, 2021; Raman et al., 2022) such as address. The architecture consists 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 ends-of-sequence token. The decoder's outputs are forwarded to the linear layer, of which the number of neurons equals 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.

Deepparse proposes four pre-trained tagging model architectures: one using each embedding model approach, namely **fastText** and **BPEmb**, and one using each embedding model approach with an added attention mechanisms. 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 (Bahdanau et al., 2015) since they have been shown to improve models' performance and help address some of the issues RNNs suffer from when dealing with long sequences. Also, Yassine et al. (2022) has shown that the attention mechanism has significantly increased performance for incomplete addresses. Incomplete addresses do not include all the components defined by a country-written standard—for example, an address missing its postal code. They are cumbersome and cause problems for many industries, such as delivery services and insurance companies (Nagabhushan, 2009).

Choosing a Model The difference between all four models is their capabilities to generate better results on unseen address patterns and unseen language. For example, as shown in Yassine et al. (2020), BPEmb embeddings models generate better parsing on address from India, even if the language and address pattern was unseen during training compared to FastText embeddings model. However, this increase in generalization performance comes at the cost of longer inference time (will be discussed in section 4). As shown in Yassine et al.

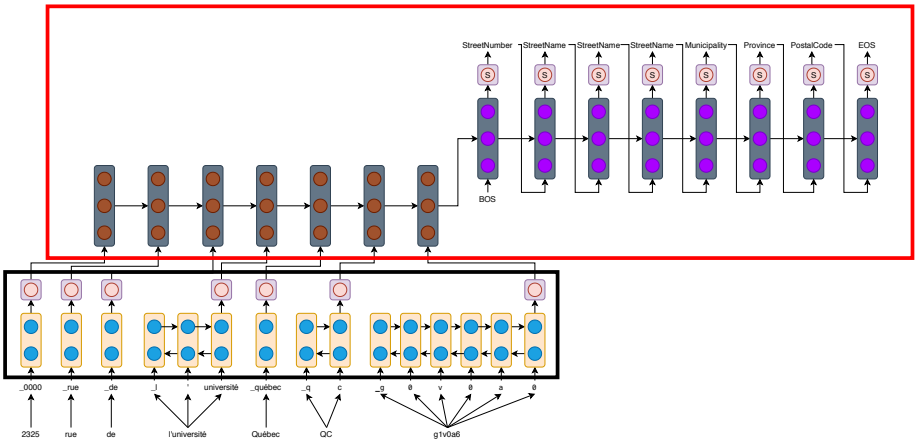

Figure 1: Illustration of our architecture using one of the two embedding model component (black) approach. Each word in the address is encoded using an embedding model, in this case, MultiBPEmb (the BPE segmentation algorithm replaces the numbers in the address with zeros). The embeddings are fed to a 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 tagging model components (red). The "S" in the fully connected layer following the Seq2Seq decoder stands for the Softmax function.

(2022), models using the attention mechanism also demonstrate the same improved generalization performance compared to their respective embeddings approaches but with the same cost of inference performance. Thus, one must trade off generalization performance over inference performance.

3.3 Developing a New Parser

One of the unique particularities of Deepparse is the ability to develop a new parser for one's specific needs. Namely, one can fine-tune one of our pre-trained models for their specific needs using our public dataset or theirs. Doing so can improve Deepparse's performance on new data or unseen countries, giving Deepparse great flexibility. As shown in Figure 2, developing (i.e. fine-tuning) a new parser using our pre-trained public models is relatively easy and can be done with a few Python lines of code.

Moreover, as shown in Figure 3, one can also use Deepparse to retrain our pre-trained models on new prediction tags easily, and it is not restricted to the ones we have used during training, making it flexible for new addresses pattern.

Finally, as shown in Figure 4 it is also possible to easily reconfigure the tagging model architecture to either create a smaller architecture, thus potentially reducing memory usage and inference time, or increase it to improve performance on more complex address data. Also, one can do all of the above at the same time.

4 Practical results

In this section, since Libpostal and Deeparse are comparable in terms of accuracy, both are almost perfect; we benchmark Deepparse memory usage and inference time with 183,000 addresses of the Deepparse dataset. Our parsing experiment processes 183,000 addresses using different batch sizes ($2^0, \ldots, 2^9$) and assesses memory usage and inference time performance for Libpostal and Deepparse. Since Deepparse can batch address, we assess the inference time as the average processing time per address (i.e. $\frac{\text{Total time to process all addresses}}{183{,}000} =$ time per address). Libpostal does not offer batching functionality. The experiment used a GPU and a CPU to assess the accelerator's gain. Thus, we also assess GPU memory usage in our experiment that uses such devices.

Our experiment was conducted on Linux OS 22.04, with the latest Python version (i.e. 3.11), Python memory_profiler 0.61.0, Torch 2.0 and CUDA 11.7 (done March 21, 2023). Our GPU device is an RTX 2080.

Table 1 and Table 2 present our experiment results using respectively a GPU device or not (i.e. CPU) with or without using batch processing. In both tables, we can see that Libpostal achieved better inference time performance. However, Deepparse still achieved interesting performance, particularly with batching that reduced by one order of magnitude the average processing time of execution.

```
address_parser = AddressParser(model_type="fasttext")
address_parser.retrain(dataset, train_ratio=0.8, epochs=5)
```

Figure 2: Code example to fine-tune our "FastText" pre-trained model on a new dataset for 5 epochs using a 80-20 % train-evaluation dataset ratio.

```
address_parser = AddressParser(model_type="fasttext")
new_tag_dictionary = {"ATag": 0, "AnotherTag": 1, "EOS": 2}
address_parser.retrain(dataset, prediction_tags=tag_dictionary)
```

Figure 3: Code example to retrained our "FastText" pre-trained model on a new dataset with new tags.

	GPU Memory usage (GB)	RAM usage (GB)	Mean time of execution (not batched) (s)	Mean time of execution (batched) (s)
fastText	∼1	∼8	∼0.0023	∼0.0004
fastTextAttention	∼1.1	∼8	∼0.0043	∼0.0007
BPEmb	∼1	∼1	∼0.0055	∼0.0015
BPEmbAttention	∼1.1	∼1	∼0.0081	∼0.0019
Libpostal	0	∼2.3	∼0.00004	∼N/A

Table 1: GPU and RAM usage and average processing time to parse 183,000 addresses using a GPU device with or without batching.

	RAM usage (GB)	Mean time of execution (not batched) (s)	Mean time of execution (batched) (s)
fastText	∼8	∼0.0128	∼0.0026
fastTextAttention	∼8	∼0.0230	∼0.0057
BPEmb	∼1	∼0.0179	∼0.0044
BPEmbAttention	∼1	∼0.0286	∼0.0075
Libpostal	∼1	∼0.00004	∼N/A

Table 2: RAM usage and average processing time to parse 183,000 addresses using only CPU with or without batching.

5 Future Development and Maintaining the Library

As our development roadmap, we plan to improve the documentation by adding a training guide on how one can develop its address parser. Also, we plan to offer new deep learning architecture that leverages more recent progress, such as a Transformer based architecture and to support more words embedding models, such as contextualized embeddings like ELMO embeddings (Peters et al., 2018). Moreover, we plan to offer a minimalist application to address parsing for coding laypersons. Finally, we aim at improving inference time performance by using recent integration of quantization technique (Cheng et al., 2018; Wu et al., 2020) in PyTorch, namely, "performing computations and storing tensors at lower bitwidths than floating point precision" (PyTorch, 2023). The library is maintained mainly by the library authors, and three to four releases are published yearly to improve and maintain the solution.

6 Conclusion

In conclusion, we have described Deepparse, an extendable and fine-tunable state-of-the-art library for parsing multinational street addresses. It is an open-source library, has over 99.9% test coverage and integrates easily with existing natural language processing pipelines. Deepparse offers great flexibility to users who can develop their address parser using our easy-to-use fine-tuning interface. Although slower than the Libpostal alternative implemented in low-level language C, Deepparse successfully parses more than 99% of address components.

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

N. Abid, A. ul Hasan, and F. Shafait. 2018. DeepParse: A Trainable Postal Address Parser. In *Digital Image Computing: Techniques and Applications*, pages 1–8.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Jian Cheng, Pei-song Wang, Gang Li, Qing-hao Hu, and Han-qing Lu. 2018. Recent Advances in Efficient Computation of Deep Convolutional Neural Networks. *Frontiers of Information Technology & Electronic Engineering*, 19:64–77.

```
address_parser = AddressParser(model_type="fasttext")
seq2seq_params = { "encoder_hidden_size": 512, "decoder_hidden_size": 512}
address_parser.retrain(dataset, seq2seq_params=seq2seq_params)
```

Figure 4: Code example to train a new model using our Seq2Seq architecture with a different configuration (i.e. encoder and decoder hidden size).

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word Translation Without Parallel Data.

Benjamin Heinzerling and Michael Strube. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 273–291.

Yi-Ting Huang, Yu-Yuan Chen, Chih-Chun Yang, Yeali Sun, Shun-Wen Hsiao, and Meng Chang Chen. 2019. Tagging Malware Intentions by Using Attention-Based Sequence-To-Sequence Neural Network. In *Information Security and Privacy*, pages 660–668. Springer.

Guozhe Jin and Zhezhou Yu. 2021. A Hierarchical Sequence-To-Sequence Model for Korean POS Tagging. *Transactions on Asian and Low-Resource Language Information Processing*, 20(2):1–13.

Armand Joulin, Piotr Bojanowski, Tomas Mikolov, Hervé Jégou, and Edouard Grave. 2018. Loss in Translation: Learning Bilingual Word Mapping with a Retrieval Criterion. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Xiang Li, Hakan Kardes, Xin Wang, and Ang Sun. 2014. HMM-Based Address Parsing: Efficiently Parsing Billions of Addresses on MapReduce. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 433–436. Association for Computing Machinery.

Shekoofeh Mokhtari, Ahmad Mahmoody, Dragomir Yankov, and Ning Xie. Tagging Address Queries in Maps Search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:9547–9551.

P Nagabhushan. 2009. A Soft Computing Model for Mapping Incomplete/Approximate Postal Addresses to Mail Delivery Points. *Applied Soft Computing*, 9(2):806–816.

Kostiantyn Omelianchuk, Vipul Raheja, and Oleksandr Skurzhanskyi. 2021. Text Simplification by Tagging. *arXiv:2103.05070*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. *arXiv:1802.05365*.

PyTorch. 2023. Quantization — PyTorch 2.0 documentation. Accessed online (30-07-2023) https://pytorch.org/docs/stable/quantization.html.

Karthik Raman, Iftekhar Naim, Jiecao Chen, Kazuma Hashimoto, Kiran Yalasangi, and Krishna Srinivasan. 2022. Transforming Sequence Tagging Into a Seq2Seq Task. *arXiv:2203.08378*.

S. Sharma, R. Ratti, I. Arora, A. Solanki, and G. Bhatt. 2018. Automated Parsing of Geographical Addresses: A Multilayer Feedforward Neural Network Based Approach. In *IEEE International Conference on Semantic Computing*, pages 123–130.

M. Wang, V. Haberland, A. Yeo, A. Martin, J. Howroyd, and J. M. Bishop. 2016. A Probabilistic Address Parser Using Conditional Random Fields and Stochastic Regular Grammar. In *International Conference on Data Mining Workshops*, pages 225–232.

Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. 2020. Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation. *arXiv:2004.09602*.

Sen Xu, Soren Flexner, and Vitor R. Carvalho. 2012. Geocoding billions of addresses: Toward a spatial record linkage system with big data.

Marouane Yassine and David Beauchemin. 2020. Deepparse: A State-Of-The-Art Deep Learning Multinational Addresses Parser.

Marouane Yassine, David Beauchemin, François Laviolette, and Luc Lamontagne. 2022. Multinational Address Parsing: A Zero-Shot Evaluation. *International Journal of Information Science and Technology*, 6(3):40–50.

Marouane Yassine, David Beauchemin, François Laviolette, and Luc Lamontagne. 2020. Leveraging Subword Embeddings for Multinational Address Parsing. In *2020 IEEE Congress on Information Science and Technology*, pages 353–360.