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, Université Laval

Introduction

We propose **Deepparse**, a Python opensource, 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 postprocessing needed. Moreover, the library supports fine-tuning with new data to generate a custom address parser.

Motivations :

- Address parsing is essential to many applications, such as geocoding and record linkage.
- Most applications are confined to academic endeavours or with little availability of free and easy-to-use open-source solutions.

Related work :

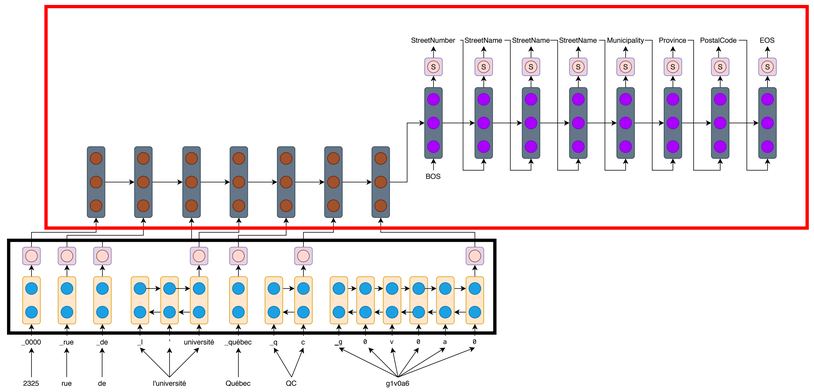
- Sharma et al. (2018) : Parse monolingual address using a feedforward neural network.
- Mokhtari et al. (2019) : Parse monolingual address using different RNN architectures.
- OpenVenues (2016) : Libpostal Python Library.

Contributions :

- We describe an open-source Python library for multinational address parsing.
- We describe its implementation details and natural extensibility due to its fine-tuning possibilities.
- We benchmark it against other open-source libraries.

Implementation Components

1. A series of simple handcrafted **pre-processing** functions to be applied as a data cleaning procedure before the next components (e.g. lowercasing).
2. An **embedding model** (black) encodes each address token into a recurrent dense representation for the overall address generated from the individual address components.
3. A **tagging model** (red) that decoded each individual address component and classified it into its appropriate tag using the address-dense representation.



Embedding Model

We proposed two pre-trained embedding

- a fixed pre-trained monolingual fastText model (pre-trained on the French language) (**fastText**),
- an encoding of words using MultiBPEmb and merge the obtained embeddings for each word into one word embedding using a Bidirectional LSTM (Bi-LSTM) (hidden state dimension of 300) (**BPEmb**).

Tagging Model

We use a Seq2Seq model consisting of

- a one-layer unidirectional LSTM encoder and a one-layer unidirectional LSTM decoder (hidden state dimension of 1024) (predict the same length of sequence),
- a fully-connected linear layer to map the representation into the tag space dimensionality (i.e. the number of tags) with a softmax activation.

The Seq2Seq decoder layer can also use attention mechanisms (self-attention).

Pre-Trained Tagging Model

Deepparse proposes **four** pre-trained models, one for each embedding model and one using each embedding model approach with added attention mechanisms.
The difference between all four models is their capabilities to generate better results on unseen address patterns and unseen language and a **trade-off between generalization performance over inference performance** (see "Practical Results" for more details).

Developping a New Parser

One of the unique particularities of Deepparse is the ability for users to fine-tune one of our pretrained models for their specific needs using our public dataset or theirs.
Fine-tuned one of our four pre-trained models

- on new data or unseen country,
- on new address components (i.e. new tags space),
- Seq2Seq architecture (i.e. hidden size, number of layers, etc.).

```
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`.

```
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).

Pratical Results

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

	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

RAM usage and average processing time to parse 183,000 addresses using only CPU 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

Conclusion

- Tackled the multinational address parsing problem with SOTA results available through an open-source Python package.
- Propose four pre-trained parsers.
- Propose a library to develop a new parser.

Future Development and Maintaining the Library :

- Improve documentation, training guide to develop a new parser.
- Develop a new Transformer pre-trained model.
- Implement contextualized embeddings for address parsing.
- Improve inference performance.