Travello: Extracting Addresses from Unstructured Text using Bi-directional Recurrent Neural Networks

Abstract

Addresses can be classified as unstructured text because they lack meta-information to be directly indexed in databases. Still they demonstrate an internal structure which can be used to automatically extract them using machine learning techniques. In this work, we describe a machine learning approach to identify addresses in unstructured text (like blogs) using Bi-directional Recurrent Neural Networks (BRNNs). We overcome the problem of lack of training data by generating synthetic free text entries and come up with problem specific features. Our system, Travello does not impose any strict condition on the structure or style of addresses leading to many applications in real life. We have released Travello and the synthetic dataset we generate publicly [1].

1 Introduction

The internet is replete with websites consisting of primarily unstructured text. Apart from official uses, the internet is also a popular place for sharing real-life experiences. Food blogs, Travelogues, etc. often contain authentic and authoritative information about famous places in a city. All such blogs, regardless of their type, talk about a place(s) of interest, give a short description along with some images, and in the end provide an address. Organizing such sources of information by extracting relevant information can be very helpful for various applications like travel itinerary managers, review websites, mapping applications, etc.

In this work, we focus on the problem of identifying address data from unstructured text like blogs, news, etc. Address data is special because it displays a loose structure within itself in the form of components like street names, city, state, phone number, zip code, etc. In informal text sources usually found on the internet, the authors do not bother to give a properly structured address like that in postal cards or business documents. At the same time, people have different styles of writing in different parts of the world. This makes the task of tagging addresses even more challenging. Recurrent Neural Networks (Rumelhart, 1986) have proven to be very effective in various NLP applications. Rather, most of the state-of-the-art solutions to problems like text translation, POS tagging, etc. are provided by RNNs. In this work, to the best of our knowledge, we give the first application of RNNs to the problem of address extraction.

2 Related Work

Originally, all approaches to address segmentation used hand-coded rule-based methods coupled with a database of cities, states, zip codes, etc. (Borkar et al., 2000) gave the first solution to automatically extracting structure from free text addresses using Hidden Markov Models (HMMs) (Rabiner and Juang, 1986). Although their approach managed to extract structure from addresses with high precision (88.9%) and recall (88.9%) on student addresses, it works only with a collection of pure address data. Such a solution will not be able to identify and extract address data from unstructured text sources like news or blog data.

(Loos and Biemann, 2008) present a statistical approach using Conditional Random Fields (CRFs) (Lafferty et al., 2001). CRFs are a generalization of HMMs which output confidence measure of the output prediction. The authors trained their model on 400 Web sites manually annotated with address information. They achieved an average precision of 0.89 and an average recall of 0.64 for the single attributes.

The above-mentioned approaches are competent in the scenarios that they describe but none is fit for extracting general addresses from informal, freely written and unstructured content like news or blogs. In this work, we describe a robust model based on Bi-directional Recurrent Neural

[1] https://github.com/kgarg8/Travello-NLP_P3

Networks (BRNNs) (Schuster and Paliwal, 1997) which can extract addresses of various formats. Also, the above works have used manually annotated datasets for training. Since RNNs need a large amount of training data, we manually synthesize our own training data to imitate blog posts found on the internet.

3 Structure of Addresses

Since we segment a web page into sentences, our task reduces to the problem of classifying a sentence as an address or non-address type. An address has many components like House/Shop name, Street name, city, country, zip code, phone number, etc. All of the above features may or may not be present in addresses that are not written according to any convention. We classify addresses according to the placement of these components in the address structure.

3.1 One-line Addresses

Many people write the complete addresses in a single line e.g. `ArtBar, Monday, July 4, 40 Edwin Land Blvd., Cambridge, 617-806-4122,artbarcambridge.com.` from Boston Magazine. These types of addresses are relatively easier to tag.

3.2 Multi-lined addresses

Multi-lined addresses are more common as compared to single line addresses. The formal order of writing an address is to mention the **Company name**, **Street Number**, **City**, **County**, **Postal Code** and **Country** but seldom do we find that such a format is followed on the internet. A typical multi-line address may look like this:-

```
6oz Espresso Bar
20 McCallum Street, #01-K1,
Singapore 069046
Tel: +65 6509 6602
Mon to Fri: 7am - 5.30pm
Nearest Station: Tanjong Pagar
```

from ladyironchef, a Singapore based food blog

So, although the address might not contain all the components that must formally be present, we observe that the ordering between the components remains unchanged i.e. Street Number comes before City name, etc. Thus, we can treat the multi-line addresses as time series data and use Recurrent Neural Networks (RNNs) to perform a sequence tagging operation.

4 Methodology

4.1 Generating the Dataset

Deep Neural Network (DNN) (LeCun et al., 2015) generally requires a large dataset for effective training (Glorot and Bengio, 2010). Since it is not feasible to manually tag so many addresses in the blogs for training, we generate our own training data to imitate a typical blog post which describes multiple places of interest as follows:

- Select a random number of English sentences taken from a repository (e.g. a book) distributed normally with a user specified mean and standard deviation.
- Create a synthetic address as follows and append it to the running text:
    1. Generate the address template, randomly selecting components with fixed probabilities (since an address may not necessarily have all the above described components e.g. Street name, City, etc.)
    2. Populate the template with random entries chosen from the database to form the imitation address.
- Repeat until required

We use the Walmart-full address dataset (wal, 2013) as our address repository. By randomly dropping address components in the template, we emulate addresses found in real-life data sources. This makes the classifier more robust to recognizing different address formats that may arise in real life situations. For one-line addresses, we append the address in the same line instead of going to the next line as in hierarchical addresses.

4.2 Features

Since address components like street names, city names, state names, country names, etc. are not arbitrary, we decided to make a database of all these components. OpenStreetMap (Ope, 2020 (accessed Novemeber 24, 2020) provides a collection of all addresses in the United States and the world. We extract all the street names from these addresses to make a custom database released publicly [2]. Using regular expressions, we identify phone numbers of various countries. A weight is assigned to every address component depending on the frequency of

[2] https://github.com/kgarg8/Travello-NLP_P3

occurrence in its respective database. The presence of phone numbers is indicated by a binary variable. We append all the weights of various address components to form a 9-dimensional feature vector for a sentence.

4.3 Network Architecture

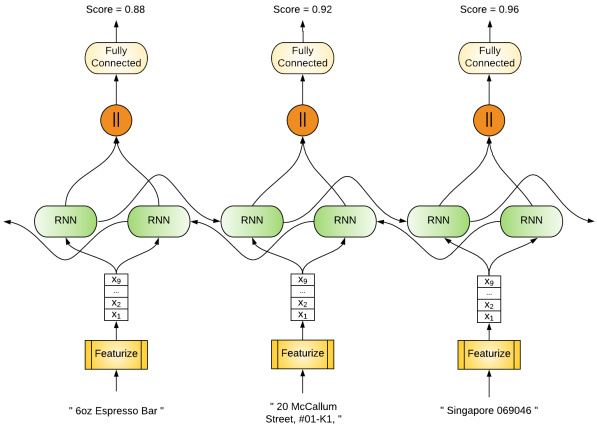

Figure 1: The Model

We use BRNNs as they can understand the context by observing the past and future simultaneously. In our application, this is specifically important as addresses are generally preceded and succeeded by non-address sentences and within addresses also we find a hierarchical structure as described above. The best performance is achieved by a simple model having one forward and one backward layer. After concatenating their results and capping with a $tanh$ dense layer on top, we get favorable results.

4.4 Identifying Address Sentences

Our model returns a score associated with a sentence that describes how likely it is an address. We found that it is not always possible to fix a hard threshold value to demarcate between an address and a non-address sentence. To mitigate this problem, we cluster the sentences into two groups using standard clustering algorithms based on the scores assigned by the model. Thus all address sentences are clustered in one group while the non-address sentences are separated into another cluster.

5 Experiments

In this section, we evaluate the performance of Travello on the following dataset. We use 3 versions LSTM models in our experiments. Version 1 uses only the 8 hand-made features. Version 2 uses only Word-to-vec embeddings (of length 128) and version 3 uses both the word embeddings and the handmade features as features.

Dataset: Since there is no publicly available, annotated dataset of text data with intermittent addresses, we create our own synthetic dataset to mimic real, unstructured data i.e. how we expect to find it in the real life. Our dataset has 125580 instances with approximately 10% of them being address dataset.

Evaluation Metrics: All the methods output a binary label for each sentence indicating whether it is an address sentence or not.

5.1 Experimental Setup

All experiments are carried out on a system with Tesla $K80$ GPU with 12GB RAM and RTX $2080Ti$ with 10GB RAM. We implement all the code in PyTorch. All the experiments, unless explicitly specified, are performed 3 times for each parameter group, and the median values are reported.

5.2 Hyper parameters

We train each model for 10 epochs. Each model has 3 hidden layers with 8 cells. We set the learning rate to 0.001 with batch size 16 and sequence length 4. We arrive at these hyper parameters after extensive parameter searching.

5.3 Results

We report the F1 scores, Precision and Recall on the test dataset (generating by splitting the master dataset in a 70:30 ratio) in Table 1. We observe that LSTM v2 and v3 perform significantly better than LSTM v1.

Table 1: Results

	LSTM v1	LSTM v2	LSTM v3
F1 Score	0.631	0.768	0.769
Precision	0.611	0.767	0.768
Recall	0.689	0.769	0.770

6 Conclusions

We present an approach to extract addresses from unstructured, free text like blogs using Bi-directional Recurrent Neural Networks. We also

describe the data generation process along with the suitable features to train the model on.

References

2013. Walmart store dataset. Last accessed 30 Aug. 2018.

2020 (accessed Novemeber 24, 2020). *OpenStreet*.

Vinayak R. Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. 2000. Automatically extracting structure from free text addresses. *IEEE Data Eng. Bull.*, 23(4):27–32.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Berenike Loos and Chris Biemann. 2008. Supporting web-based address extraction with unsupervised tagging. In *Data Analysis, Machine Learning and Applications*, pages 577–584. Springer.

Lawrence Rabiner and B Juang. 1986. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16.

DE Rumelhart. 1986. Hinton and williams, rj (1986):"learning internal representations by error propagation". *parallel distributed processing*, 1.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681.