A Geo-Tagging Framework for Address Extraction from Web Pages

Julia Efremova(✉), Ian Endres, Isaac Vidas, and Ofer Melnik

HERE Technologies, Amsterdam, The Netherlands
{julia.efremova,ian.endres,isaac.vidas,ofer.melnik}@here.com

Abstract. Searching for locations in web data and associating a document with a corresponding place on the map becomes popular in user's daily activities and it is the first step in web page processing. People often manually search for locations on a web page and then use map services to highlight them because geographic information is not always explicitly available.

In this work, we present a geo-tagging framework to extract all addresses from web pages. The solution includes an efficient web page processing approach, which combines a probabilistic language model with real-world knowledge of addresses on maps and extends geocoding services from short queries to large text documents and web pages. We discuss the main problems in dealing with web pages such as: web page noise, identification of relevant segments, and extraction of incomplete addresses. The experimental result shows precision above 91% which outperforms standard baselines.

1 Introduction

Web pages and text documents often contain geographical information mentioned as free text. We consider under geographic information the location of a real place, for instance a theater, a shop or a restaurant. Locations typically occur in a document in the form of unstructured descriptions instead of specified geo-coordinates with a point on the map and they are difficult to identify compared to structured data.

The importance of location extraction is the possibility to associate them with the exact map coordinates: latitude and longitude. This process is called geocoding. Available geo-information increases the popularity of a place and can be used as a part of an advanced content analysis. Locations are widely used by recommendation systems which often display the most relevant places with specifically mentioned locations and ignore free text.

In this work, we focus on geo-tagging of web pages and our goal is to identify locations in a document and to assign them appropriate geotags. Geo-tagging is common in social media, for instance, Twitter posts, Flickr pictures, etc. often contain geotags.

Dealing with web pages is different from identifying locations in structured documents and in short text queries, due to web page heterogeneity, lack of

P. Perner (Ed.): ICDM 2018, LNAI 10933, pp. 288–295, 2018.
https://doi.org/10.1007/978-3-319-95786-9_22

structure and a high level of noise. They require special pre-processing techniques to reduce the amount of irrelevant information and to discover targeted parts.

In this work, we design a geo-tagging framework for address extraction from web pages. The main distinction of our work is a designed framework for full text geocoding, which combines state-of-the-art techniques from various fields (web page processing, information extraction and machine learning), extends standard address extraction solutions to the domain of web pages.

The contribution of this paper is a designed framework for address extraction which has the following advantages:

1. highlighting of the main aspects of web page processing such as: web page segmentation and noise reduction;
2. augmenting a probabilistic language based model with geographic information;
3. validating addresses by real-world knowledge of maps on top of the process;
4. resolving similar and partial addresses (address resolution) by ranking results;
5. extending address extraction from short queries large text documents and web pages.

2 Related Work on Address Extraction from Web Pages

The problem of address extraction has been studied by various researchers. Recently, many approaches to address extraction from free text have been proposed. Most of them belong to the natural language processing fields and include techniques from sentence breaking and part-of-speech tagging to content parsing and semantic analysis [3].

Yu [10] presented address extraction using rule-based, machine learning and hybrid techniques based on decision tree classifier and n-gram model. He uses regular expressions combined with lexical features such as: punctuation, initial capital, contain digits, etc.

Chang and Li [2] proposed the MapMarker framework where they adopt conditional random fields (CRF) for web page segmentation and solve address extraction problem by sequence labeling. Their approach relies on the quality of training data and the learned language model.

Melo and Martins [7] prepared a survey where they predict the geo-coordinates for a document using the whole textual content. They show early approaches based on rule-based heuristics; machine learning approaches including feature selection and classification; and extended approaches using place disambiguation and coordinate estimations. The difference of our approach in assumption that a given document can contain multiple locations and we verify identified addresses by searching them on the real map.

Another existing solution for address parsing and expansion is Libpostal[1] which is an open source library trained on a large dataset of real-world addresses

[1] https://mapzen.com/blog/libpostal.

from OpenStreetMap[2] [4,5]. Libpostal parses addresses by searching for house numbers, roads, cities, etc. For our purposes, the main shortcoming of Libpostal is the lack verification whether an address really exists on the map without identification of empty requests.

The distinction of our work comparing to prior efforts is in applying geocoding service on top of the whole extraction chain and use it as the main address validator. Another distinction is in address resolution by ranking similar addresses in the same neighborhood and final identification of complete and verifies addresses.

3 Data Description: A Collection of Web Pages

We use a collection of crawled and cashed web pages provided by *Common Crawl* [8]. Every web page is available in the WARC, WAT and WET formats which stand for raw web page data, metadata and extracted text. Common Crawl is a public corpus, mostly stored on Amazon Web Services[3].

A subset of the CommonCrawl dataset has schema information in the microdata format with manual annotation of the main content such as: addresses, names, phones, as well as micro content such as: streets, regions, postal codes, etc. Microdata are a set of attributes which are embedded into standard HTML tags as a special property *itemprop*.

Table 1. An example of microdata annotation. The annotated content is in bold

<DIV class="column threeColumn">
<DIV itemprop=**"address"** itemtype="http://schema.org/PostalAddress">
<SPAN itemprop=**"streetAddress"**>505 Grand Rd</SPAN>
 <SPAN itemprop=**"addressLocality"**>East Wenatchee< /SPAN>
<SPAN itemprop=**"addressRegion"**>WA< /SPAN>
<SPAN itemprop=**"postalCode"**>98802< /SPAN><BR/ >< /DIV>
<DIV><A href="/restaurateurs/your-business/6023/">
Is this your business?</A></DIV>

Table 1 is an example of detailed address annotation (street, locality, region and postal code) embedded into the DIV and SPAN tags. Documents with microdata annotation are a great source for training and evaluation purposes. According to the W3Techs survey only around 12.7% web pages have microdata tags compared to the entire web; hence, there is a need for automatic extraction. Microdata is an excellent source of ground truth, which can be used for training and evaluation purposes. It provides structured data and makes web pages easier to read instead of dealing with free text. Particularly, we use the *Restaurant* schema[4] from Schema.org of microdata annotations.

[2] https://osmnames.org.
[3] https://aws.amazon.com/public-datasets/common-crawl/.
[4] http://schema.org/Restaurant.

4 An Introduced Geo-Tagging Framework

Our address extraction framework includes the following steps: *text extraction*, *tokenization*, *annotation by geographical features*, *annotation by lexical features*, *sliding window generation*, *window classification*, *window geocoding* and *address ranking* illustrated in Fig. 1.

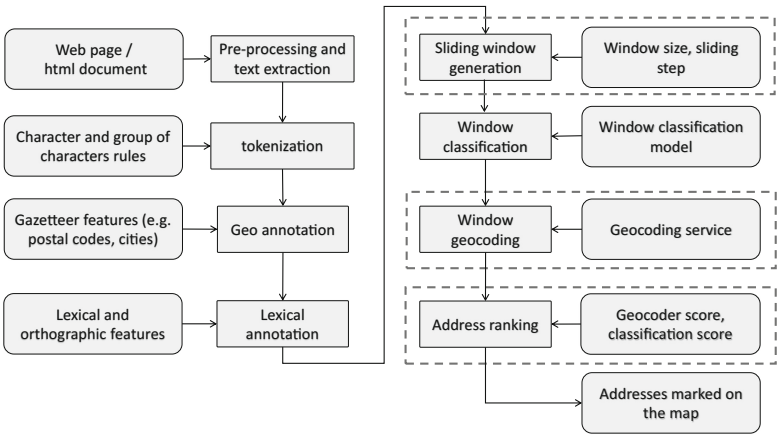


Fig. 1. Components of an introduced address extraction framework

In the beginning, a web document is cleaned and preprocessed in order to reduce HTML noise and to convert a raw document into an informative representation.

Then in the *tokenization* step, data is divided into small elements called tokens (words, bi-grams, tri-grams, etc.) based on a certain splitting criteria.

In the third and the forth step, we annotate tokens by different groups of features: lexical described in [10] which indicate whether a token is a digit, contains a digit, is a title, is capitalized, starts with a capital letter; and geographical features, which show if a token is a state, is a postal code, is a city, etc. The *Geonames*[5] [1] project and the *Libpostal* library discussed in Sect. 2 can be used as a source of geo-features.

Subsequently, term frequency transforms feature annotation into numerical feature vectors. It calculates the number of times each feature occurred.

Afterwords, a sliding window of a fixed size and with a fixed sliding step moves over annotated tokens from top to bottom. A approach based on sliding windows allows to analyze the entire web page content and does not require prior web page segmentation or other pre-processing. A common solution for this step is web page segmentation, for instance by Latent Semantic Indexing [6]. LSA identifies semantically close segments by associating related words, however it

[5] http://www.geonames.org.

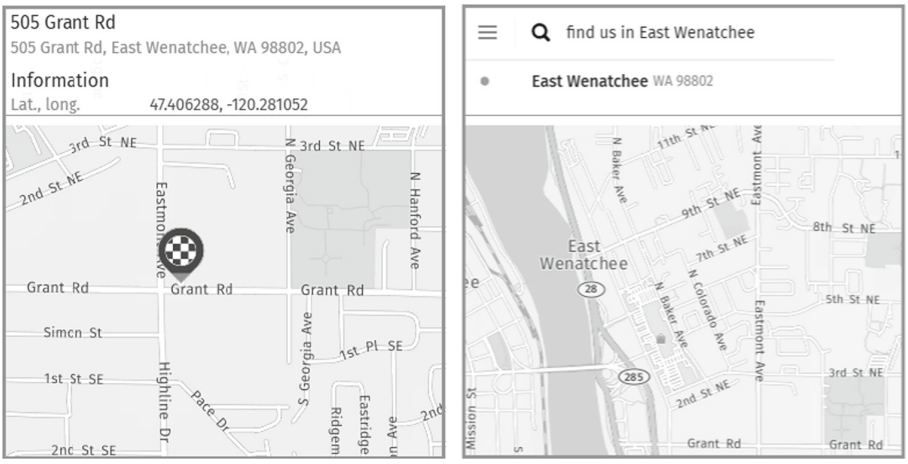
leads to missing addresses due to incorrect segments which we aim to avoid. Table 2 illustrates sliding windows for the piece of text *'Papa Murphy's East Wenatchee 505 B Grant Road East Wenatchee WA, 98802'* in the example in Table 1.

Table 2. An illustration of sliding windows

w_1	Papa Murphy's East Wenatchee *505 B Grant Road East Wenatchee*
w_2	Murphy's East Wenatchee *505 B Grant Road East Wenatchee WA*
w_3	East Wenatchee *505 B Grant Road East Wenatchee WA, 98802*

After creating a sliding window, a binary classifier computes a classification score for each window and identifies address candidates. We choose a *Support Vector Machines* (SVM) classifier, since it is a widely-used robust classifier [9]. Classification requires a prior training phase on an annotated dataset to make a prediction on new data. We consider annotated addresses in the dataset discussed in Sect. 3 as positive examples and windows without addresses as negative.

Then, the Geocoder validates a candidate window by searching its geo-coordinates on the map. We use HERE Geocoder[6] which provides a powerful service for address parsing and validation. Figure 2a illustrates address validation by searching it on the map.



(a) an example of a compete address search (b) an example of a partial address search

Fig. 2. An illustration of address verification by searching it on the map

An exact point location could not be identified on the map in case of partial addresses. Geocoder then refers to a general area as shown in Fig. 2b.

[6] https://developer.here.com.

The last step is addresses ranking to solve the address disambiguation problem. A web page can contain the same address mentioned multiple times, address variations and parts of the same address. As an example consider the two addresses: *'505 Grand Rd East Wenatchee'* and *'East Wenatchee'*, the second refers to a region of the first place. We use a Geocoder relevance score and a classification score to solve the problem of similar addresses and to find final and the most detailed and verified addresses on a web page.

5 Experimental Results

5.1 Applying a Geo-Tagging Framework

Before applying the designed framework, it is important to define a window size and a sliding step based to the two main criteria:

1. a window size should fit a complete address;
2. a window size should have a minimum number of irrelevant tokens (noise);
3. a sliding step should not be large otherwise parts of the same address will belong only to neighboring windows.

We learn an optimal window size from length distributions of annotated addresses presented on Fig. 3. The typical address length is between 5 to 10 tokens in microdata annotation and between 6 and 10. We choose the window size of 10 tokens which is an optimal size for most of the addresses.

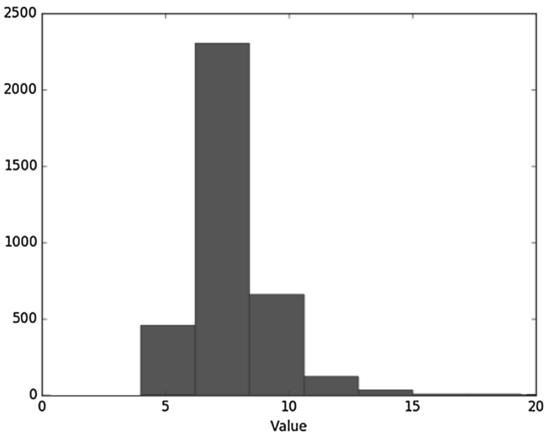


Fig. 3. Distribution of address length in tokens. X-axis represents the address length and Y-axis represents the number of addresses analyzed

5.2 Result Evaluation

We conduct experiments on the annotated datasets described in Sect. 3. In order to assess the performance of our results, we apply 10-fold cross-validation. We randomly partition the annotated data into 10 equal size subsets. Then one subset is chosen as the validation data for testing the classifier, and the remaining subsets are used for training purposes. Then the process is repeated 10 times, with each of the 10 subsets used exactly once as the validation dataset.

We use regular expressions (RE) as the first baseline and learn the RE grammar on a training set considering, for instance text between a house number (represented by digits) and a postal code as an address.

The second baseline is based on segmentation by key words. The Latent Semantic Analysis (LSA) introduces in Sect. 4 searches for relevant segments similar to a list of keywords obtained by topic modeling or from a relevant vocabulary. In our case, we use GeoNames database as a source of initial geo-keywords.

When segments are identified, we search for the full address only within the identified segments. This method assumes that an address should contain keywords (e.g. cities, postal codes) and it is not robust in case of misspellings, for instance comparing *'San Francisco'* and *'SF'*.

Table 3 shows comparative evaluation of the designed geo-tagging framework and two baselines in standard metrics: precision and recall. Address extraction by RE has the lowest performance since addresses can occur in any format which is differ from RE. The baseline 2 demonstrates almost double improvement in performance however recall is only 50% due to missed address segments. The proposed approach outperforms the baselines in terms of precision and also recall which is improved up to around 92%.

Table 3. Evaluation of the geo-tagging framework for address extraction

Address extraction	Precision	Recall
Baseline 1: regular expressions	47.7%	33.61%
Baseline 2: LSA segmentation with subsequent classification	80.67%	53.61%
The proposed geo-tagging framework	91.12%	92.86%

Initial error analysis of incorrectly extracted addresses revealed common issues. One of which is confusions between the main address elements, for instance place names and street names, house numbers and other numbers in a window. If a number (which is not an address part) occurs in a window, it can be matched it to a house number and affect address extraction.

Therefore, identification of precise address boundaries can be a potential extension of this work. One solution is to learn an address grammar. For instance, a postal code can be used as an indicator of the ending boundary.

Probabilistic approaches based on sequence annotation such as conditional random fields is another way to find address boundaries, however it might reduce recall and lead to missing addresses because it does not take into account real map knowledge.

6 Conclusion

In this work, we introduced a geo-tagging framework for address extraction from web pages. We addressed the main challenges in working with web data such as: structural heterogeneity, web page pre-processing, segment analysis and address validation. We incorporated sliding windows and avoided web page segmentation in order to improve recall and to save precision on the high level. We discussed in detail different groups of feature extraction and also sources to obtain geo-information. We showed how to validate extracted addresses by analyzing their existence on the map and applied geocoding on the top of machine learning models.

It is possible to extend the designed framework in various directions, for instance by adding visual features, by incorporating the DOM structure or defining a window size dynamically instead of having it fixed.

References

1. Ahlers, D.: Assessment of the accuracy of geonames gazetteer data. In: Proceedings of the 7th Workshop on Geographic Information Retrieval, GIR 2013, pp. 74–81. ACM, USA (2013)
2. Chang, C.-H., Li, S.-Y.: MapMarker: extraction of postal addresses and associated information for general web pages, pp. 105–111. IEEE Computer Society (2010)
3. Gupta, V., Lehal, G.S.: A survey of text mining techniques and applications. J. Emerg. Technol. Web. Intell. **1**(1), 60–69 (2009)
4. Haklay, M., Weber, P.: Openstreetmap: user-generated street maps. Pervasive Comput. **7**(4), 12–18 (2008)
5. Lawrence, C., Riezler, S.: NLmaps: a natural language interface to query OpenStreetMap. In: COLING, Demos, pp. 6–10. ACL (2016)
6. Li, H., Xu, J.: Semantic matching in search. Found. Trends Inf. Retr. **7**(5), 343–469 (2014)
7. Melo, F., Martins, B.: Automated geocoding of textual documents: a survey of current approaches. Trans. GIS **21**(1), 3–38 (2017)
8. Meusel, R., Petrovski, P., Bizer, C.: The webdatacommons microdata, RDFa and microformat dataset series. In: Mika, P., et al. (eds.) ISWC 2014. LNCS, vol. 8796, pp. 277–292. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11964-9_18
9. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques, 4th edn. Morgan Kaufmann Publishers Inc., Burlington (2016)
10. Yu, Z.: High accuracy postal address extraction from web pages (2007)