Tagging Address Queries in Maps Search

Shekoofeh Mokhtari,[†] **Ahmad Mahmoody,**[‡] **Dragomir Yankov,**[‡] **Ning Xie**[†]
smokh004@fiu.edu, ahmahmoo@microsoft.com, dragoy@microsoft.com, nxie@fiu.edu
[†]Florida International University, Miami, Florida
[‡]Microsoft Corporation, Sunnyvale, California

Abstract

Map search is a major vertical in all popular search engines. It also plays an important role in personal assistants on mobile, home or desktop devices. A significant fraction of map search traffic is comprised of "address queries" - queries where either the entire query or some terms in it refer to an address or part of an address (road segment, intersection etc.). Here we demonstrate that correctly understanding and tagging address queries are critical for map search engines to fulfill them. We describe several recurrent sequence architectures for tagging such queries. We compare their performance on two subcategories of address queries - single entity (aka *single point*) addresses and multi entity (aka *multi point*) addresses, and finish by providing guidance on the best practices when dealing with each of these subcategories.

Introduction

Map search has become an integral part of our everyday experience - users search for locations to visit or just for information. On mobile devices Maps are one of the highest downloaded and used apps (Comscore 2017). One of the main categories of queries which map search handles is "address queries". We use this term loosely to indicate a broad set of address patterns - cases where either the entire query or part of it contain address reference. The address reference itself can constitute a complete *point address* or *partial* reference to such. Some common address query patterns which users issue and expect map search to resolve are:

- {*123 Main St, San Francisco, CA 94105*} - complete point address query.
- {*30 Rockefeller Plaza*}, {*350 5th Ave, New York*} - partial queries, with city, post code or other information missing.
- {*Pennsylvania Ave Washington DC*}, {*Fremont, Seattle*} - road or neighborhood queries, missing exact address.
- {*Quai Branly et Avenue de la Bourdonnais*} - intersection queries.
- {*Bakery near Castro Street, Mountain View*} - business or place with road as location reference. Etc.

Address queries are fulfilled by a special search engine known as *geocoder* (Berkhin et al. 2015). Similar to web

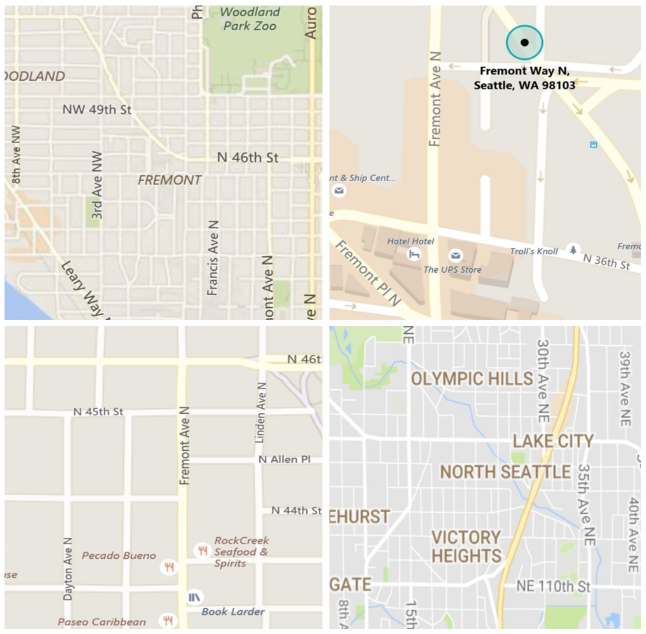

Figure 1: Depending on the query interpretation the geocoder may infer different results: *Top* Possible results for the query: {*Fremont Seattle*}. *Bottom* Possible results for the query: {*Fremont Ave North Seattle*}

search engines, geocoders map the terms of the query to certain documents, in this case known as *geo entities* which are subsequently formatted and returned as results with their associated latitude and longitude. Unlike web search however, geocoders are expected to return very limited (usually only one) results with very high precision. The cost to users for returning a bad result can be very high. For example, they may use the result as driving direction and end up in a wrong place and incur time or other loss. Therefore, having accurate interpretation of address queries is of critical importance to geocoding and map search in general.

Tagging the terms in an address query correctly turns out to be a challenging problem. On one hand, it is very dependent on the user language and market, and on the other hand, even in the same language there might be different query interpretations that lead to different geocoded results.

The interpretations will also impact how confident is the geocoder in the identified result (Berkhin et al. 2015). This is important because geocoding confidence is often used by consumer system, e.g. navigation systems, when deciding whether to accept the result or to inform the user that no location was found matching their query which leads to decrease in recall and poor user experience. As illustration consider the following two queries:

Query 1 : $\{$*Fremont Seattle*$\}$. There are several possible annotations for this query - 1) Neighborhood: Fremont, City: Seattle; 2) Street name: Fremont, City: Seattle; 3) Business: Fremont (e.g. colloquial for Fremont Brewery), City: Seattle. Depending on which interpretation we assume most likely, the geocoder may infer different results with different confidence (in the second interpretation the confidence may be lower as there are multiple equally likely candidates for the road “Fremont” - Fremont Way, Fremont Pl, and Fremont Ave) - see Fig 1 top.

Query 2 $\{$*Fremont Ave North Seattle*$\}$. Interpretations: 1) Street: Fremont Ave North, City: Seattle; 2) Street: Fremont Ave, Neighborhood (colloquial): North Seattle (see Fig 1 bottom). In the second interpretation the geocoder will not find a road with such name in such neighborhood and may simply return the up-hierarchy result of North Seattle.

Hence, it is central to correctly tag each token of address queries. There have been a few strands of works focusing on tagging address related queries (Wang et al. 2016; Li et al. 2014; Churches et al. 2002; Borkar, Deshmukh, and Sarawagi 2001), however traditional rule-based address parsers (Churches et al. 2002) require domain knowledge and have been shown to be limited in classification accuracy. Probabilistic methods such as (Wang et al. 2016; Li et al. 2014) which are based on HMM and CRF have been developed to improve the rule based methods but they have some difficulties dealing with rich and complex features. In recent years RNN based models have shown state of the art results on sequence tagging problems (Ma and Hovy 2016). In this work, after formalizing the address tagging problem, we outline several recurrent architectures suitable for modeling it. We analyze their performance on two structurally different query patterns that we find prevalent in map search logs. We conclude with guidance on the practices for choosing a suitable architecture when working with such patterns.

Tagging Address Queries

Analyzing large data sets with address queries from an industrial map search engine, we observe certain aspects which make correctly tagging such queries a non-trivial task. To enumerate a few of the challenges:

- Data is unstructured (or semi-structured), with irregularities or omissions.
- Data is noisy and may have typos and abbreviations.
- The problem is market[1] dependent, with large number of tags present in each market (see Table 1 for en-US).
- Small number of head terms and large number of tail terms.
- Data is sparse. Unlike web search queries, most address queries appear only once in the logs.

[1]A market is defined by language and country, e.g. fr-CA are French queries issued in Canada.

Abbr.	Tag Description
HN	House Number
SBT	Sub address Type (Building, Tower)
BN	Building Name
SD	Street Direction
ST	Street Type (Ave, St, etc.)
SN	Street Name
CI	City
CO	Country
ST	State name
N	Neighborhood (Kirkland)
ZP	Zip Code
OT	Occupancy Type (Floor, Suit, Apt)
SP	Separator (near, by, in , etc.)
B	Business (Starbucks,Walgreen’s, etc.)
UNK	Unknown (Not related to address)

Table 1: Abbreviation and description of existing tags in both multi point and single point address queries.

The above points are further exacerbated by the shift that conversational interfaces in personal assistants provide (e.g. Cortana, Google Now, Siri, Alexa etc.) - users start issuing increasingly longer and more colloquial queries. So not only are the queries sparse and form a long tail, but also contain multiple irrelevant terms which need to be identified and tagged correctly.

To build a system that learns to interpret correctly address queries we first focus on understanding how many distinct geo-entities are present in them. In doing so we identify two types of queries: Single point (SP) - queries that contain a single entity (point), and and Multi point (MP) - queries that contain references to multiple geo-entities (points). We now go into more details for each of these categories.

For convenience, in Table 1 we summarize some of the tags that our models are trained to identify. The table only shows a limited number of tags. In en-US addresses judges identified more than 20 tags.

Single Point Queries

Single point address queries have standard format which is mostly used by national postal service of each country. The format of the address queries greatly depends on the country. Figure 2 shows an example of a SP address query for United States. The query contains only one geo-entity, in this case a fully-qualified complete address point.

Multi Point Queries

MP queries contain terms that identify multiple geo-entities. Some of them define the expected result and others define points of reference. For example, many MP queries follow the pattern:

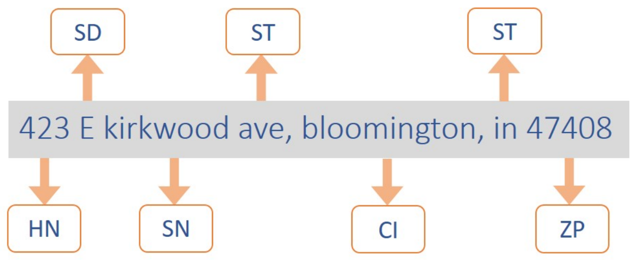

Figure 2: Example of a Single Point query. The result (and the query) identify one address point entity. One or more terms may be missing from the query.

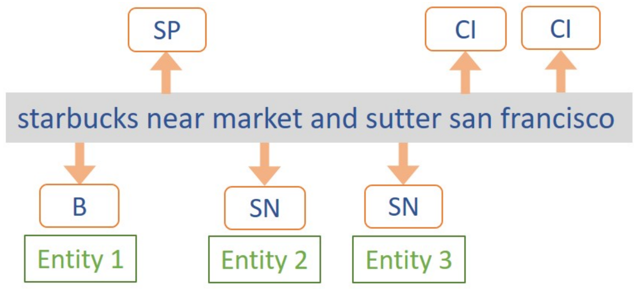

Figure 3: Example of a Multi Point query. The *where* part after the separator can be a full formed or partial address address, or it can itself contain multiple entities.

[what][separator][where]

in which the "[where] part" is used as a reference. Referential queries are a very common way through which users specify addresses in some countries, e.g. India (Berkhin et al. 2015). Figure 3 shows one such MP query where a user is asking about a particular "Starbucks" (entity 1- business) that is close to an intersection (reference point). The intersection is comprised of two geo-entities - "Market Str" (entity 2) and "Sutter Str" (entity 3) in San Francisco. Intersections themselves are another common MP query pattern.

Sequence Tagging Architectures

Sequence tagging is a well-studied task in NLP including named entity recognition (NER), chunking, and part of speech tagging (POS). Most of the existing approaches are probabilistic in nature such as Hidden Markov models (HMM), Maximum entropy Markov models (MEMM) (McCallum, Freitag, and Pereira 2000), and Conditional Random Fields (CRF) (Lafferty, McCallum, and Pereira 2001). There are several neural network based approaches to address the sequence tagging task (Collobert et al. 2011; Ma and Hovy 2016).

In this work we experimented with three categories of architectures:

One-Directional This general structure consists of (i) an embedding layer, (ii) a forward recurrent cell, and (iii) a fully connected layer (Kawakami 2008). For choosing the recurrent cell, we have two choices, and each results in a different architecture:

- **Forward-RNN:** The one-directional architecture that uses the vanilla RNN as its recurrent cell.
- **Forward-LSTM:** The one-directional architecture that uses the LSTM as its recurrent cell.

Bi-Directional Our general architecture for bi-directional structure consists of (i) an embedding layers of words, (2) a forward recurrent cell applied on the input sequence, also (3) a backward recurrent cell applied on the input sequence, and (4) fully connected layer applied on the concatenation of the forward and backward recurrent cells (Kawakami 2008). Similar to the one-directional structure, we have two choices for the recurrent cells, and each provide a different architecture:

- **Bi-RNN:** The bi-directional architecture that uses the vanilla RNN (Pineda 1987) as its recurrent cell.
- **Bi-LSTM:** The bi-directional architecture that uses the LSTM (Hochreiter and Schmidhuber 1997) as its recurrent cell.

Sequence-to-sequence We also try the sequence-to-sequence model introduced in (Hochreiter and Schmidhuber 1997) that has shown great success in neural machine translation (NMT), speech recognition, and text summarization. (Sutskever, Vinyals, and Le 2014).

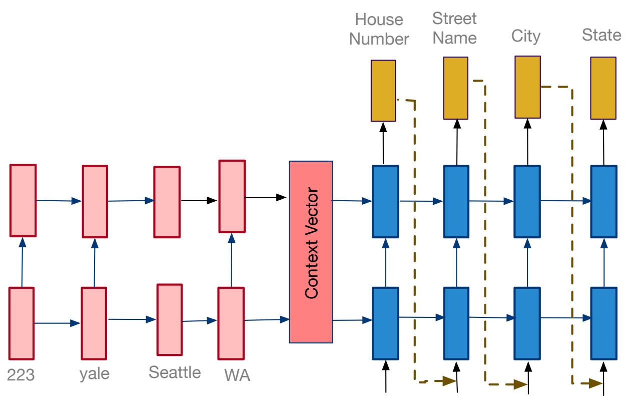

Figure 4: Sequence to Sequence model architecture for sequence tagging task. The light red boxes are LSTM encoder and The blue boxes are LSTM decoder.

Experiments

For training the models, we implement all the models in Microsoft cognitive toolkit (CNTK) 2.0[2]. All the experiments are performed on a single GPU machine. We run all the models up to 256 epochs and select the model that achieves the best accuracy on the validation set. We use hidden size h = 128 and Optimization is carried out using *Adam*, with a fixed learning rate of 0.1. For training our neural networks,

[2]https://github.com/Microsoft/CNTK

	en-US(SP)	en-US(SP & MP)	Yelp SP	Yelp MP
#Train	180K	770K	100K	1M
#Validation	12K	180K	2.5K	100k
#Test	42K	50K	4K	200K

Table 2: Data statistics of the real and synthetic datasets.

Model	en-US(SP)	en-US(SP & MP)	Yelp(SP)	Yelp(MP)
F-RNN	89.46	72.16	96.97	98.24
Bi-RNN	98.48	96.14	**98.44**	99.68
F-LSTM	90.09	73.02	97.09	98.23
Bi-LSTM	98.77	96.69	98.39	**99.69**
Seq2Seq	**99.17**	**97.50**	98.22	**99.69**

Table 3: Per query tagging test accuracy of all models on Yelp and en-US datasets. en-US is real English address queries from United States. Yelp SP and Yelp MP are generated address queries from Yelp dataset.

we only keep the frequent $|V| > 5$ words, and map all other words to an *UNK* token. In order to deal with segmentation problems in queries such as *Las Vegas*, we transform them into BIO encoding.

Data Collection

We conduct all the experiments on both en-US real and synthetic data. The synthetic datasets have been generated from the sheer volume of local business information available on-line in Yelp academic dataset [3]. Yelp dataset includes information about 156k local businesses from 11 metropolitan areas across 4 countries. For our experiment, we only keep the united states addresses which are about 100K local businesses. We created single point (SP) and multi point (MP) address queries from local business information such as address, neighborhood. Table 2 provides some statistics on the two datasets.

Real Data. These queries are collected form logs and labeled by human judges as SP and MP queries. We created two separate dataset including *en-US (SP)* which purely contains SP queries and *en-US (SP & MP)* which has combination of single and multi point queries in order to mimic the real queries in map search.

Yelp Single Point (SP). For generating synthetic single point address queries, We extracted the address of all available businesses in United States and employed Parserator [4] for parsing unstructured address strings into address components.

Yelp Multi Point (MP). For generating the Yelp MP queries we follow the above outlined common MP pattern:

[what][separator][where]

Let us term the entities in the [where] part primary entities and in the [what] part secondary entities. The secondary entities mostly involves business name or business category, from address fields of Yelp and the primary entity in neighborhood, city and businesses and roads near the secondary entity. We employ different patterns for generating MP queries such as business near road, business near business, business near place, road near road, etc. Then, we used some perturbation techniques in order to make the generated query looks like the real data. Data and more details will be publicly available [5].

Evaluation and Results

Since geocoders require highly accurate tags for address queries, we evaluate the performance of models per query and not per entity. We compare the performance of RNN based model such as RNN, Bi-RNN,the bi-direction RNN; LSTM, Bi-LSTM, and sequence to sequence model. Our experiments lead to novel insights and practical advice for building and extending tagging address queries. Sequence to sequence models perform really well on real data (en-US) according to the results in Table 3. For pure single point queries (Yelp SP), Bi-RNN performs slightly better than sequence to sequence model.

Our sequence to sequence model with 2 layers of LSTM in encoder and 3 layers of LSTM in decoder achieves the best performance results. Table 4 provides an example of address query and tagging result from models such as Forward LSTM and Bi-LSTM. Since forward LSTM and forward runs from left to right, the decision for each tag has no information about upcoming words. As shown in Table 4, F-LSTM Tagged *seattle* as street name and didn't detect as the end of the query but bidirectional models look ahead until the end of the sentence through a backward recurrence. Bi-LSTM still didn't tagged the query correctly but better than forward LSTM.

Conclusions

In this paper, we examined several RNN based models for tagging address queries. Our experiments show that sequence to sequence model perform well on both real and

[3]https://www.yelp.com/dataset/challenge

[4]https://parserator.datamade.us/usaddress

[5]https://github.com/smokh004/AddressQuery

Model	Query: "223 yale seattle"
F-LSTM	{*HouseNumber*, *StreetName*, *StreetName*}
Bi-LSTM	{*HouseNumber*, *Unknown*, *City*}
Correct Tagging	{*HouseNumber*, *StreetName*, *City*}

Table 4: An example of tagging address query with trained model on en-US dataset.

synthetic corpora. For future works, we are looking to also run experiments with other markets and also checking other complicated neural network architecture such as character level and hierarchical to improve the results.

Acknowledgments

The majority of this work was done when first author was interning at Microsoft. This work was supported by Florida International University dissertation year fellowship.

References

Berkhin, P.; Evans, M. R.; Teodorescu, F.; Wu, W.; and Yankov, D. 2015. A new approach to geocoding: Binggc. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '15, 7:1–7:10.

Borkar, V.; Deshmukh, K.; and Sarawagi, S. 2001. Automatic segmentation of text into structured records. In *ACM SIGMOD Record*, volume 30, 175–186. ACM.

Churches, T.; Christen, P.; Lim, K.; and Zhu, J. X. 2002. Preparation of name and address data for record linkage using hidden markov models. *BMC Medical Informatics and Decision Making* 2(1):9.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Comscore. 2017. Mobile app report.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kawakami, K. 2008. *Supervised Sequence Labelling with Recurrent Neural Networks*. Ph.D. Dissertation, Ph. D. thesis, Technical University of Munich.

Lafferty, J.; McCallum, A.; and Pereira, F. C. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Li, X.; Kardes, H.; Wang, X.; and Sun, A. 2014. Hmm-based address parsing with massive synthetic training data generation. In *Proceedings of the 4th International Workshop on Location and the Web*, 33–36. ACM.

Ma, X., and Hovy, E. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354*.

McCallum, A.; Freitag, D.; and Pereira, F. C. 2000. Maximum entropy markov models for information extraction and segmentation. In *Icml*, volume 17, 591–598.

Pineda, F. J. 1987. Generalization of back-propagation to recurrent neural networks. *Physical review letters* 59(19):2229.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Wang, M.; Haberland, V.; Yeo, A.; Martin, A.; Howroyd, J.; and Bishop, J. M. 2016. A probabilistic address parser using conditional random fields and stochastic regular grammar. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*, 225–232. IEEE.