HMM-based Address Parsing with Massive Synthetic Training Data Generation

Xiang Li[†‡] Hakan Kardes[†] Xin Wang[†] Ang Sun[†]
[†]Data Research, inome, Inc.
[‡]Computer Science Department, New York University
xiangli@cs.nyu.edu, {hkardes, xwang, asun}@inome.com

ABSTRACT

Record linkage is the task of identifying which records in one or more data collections refer to the same entity, and address is one of the most commonly used fields in databases. Hence, segmentation of the raw addresses into a set of semantic fields is the primary step in this task. In this paper, we present a probabilistic address parsing system based on the Hidden Markov Model. We also introduce several novel approaches of synthetic training data generation to build robust models for noisy real-world addresses, obtaining 95.6% F-measure. Furthermore, we demonstrate the viability and efficiency of this system for large-scale data by scaling it up to parse billions of addresses.

Categories and Subject Descriptors

I.2.7 [**Natural Language Processing**]: Text Analysis; H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithms

Keywords

Address Parsing, Large-scale Data, Record Linkage

1. INTRODUCTION

Record linkage refers to the process of joining records that refer to the same entity or event in one or more data collections. In the absence of a shared, unique key, record linkage involves the comparison of ensembles of partially-identifying, non-unique data items between pairs of records. Address is one of the most commonly used spatial data items in everyday life, and parsing addresses into semantic fields is therefore a fundamental problem for spatial record linkage (Section 2). In this paper, a fast, reliable address parser has been developed.

One of the most frequently used spatial footprints is postal address, such as "*500 108th Ave NE Suite 2200 Bellevue WA 98004-0000*". The motivation of address parsing in this paper comes directly from the demand to support people search, which aims to return individuals according to queries that use names and locations. Given that there are many people sharing the same name (e.g., there are more than 500 "Jim Smith" in California state[1]), using a combination of name and address to differentiate records is one of the essential techniques for reliable record linkage. Thus, we define a set of semantic fields in Table 1 to parse addresses, where the above sample address is parsed into corresponding fields for illustration. Since real-world addresses may contain variations or noisy information, such as wrong spellings or inconsistent orders, several approaches are applied to incorporate "noise" into the training data for better models (Section 3). Traditionally, deterministic rule-based processing systems are used to carry out this parsing procedure. This paper describes an alternative approach, the probabilistic Hidden Markov Model (HMM), which provides a viable, efficient alternative to rule-based systems (Section 4). An HMM-based system is implemented with advanced features to parse billions of raw, real-world U.S. addresses, which turns out its outstanding capacity for handling large-scale data (Section 5).

Table 1: Address Fields and Parsed Example

Field	Description	Example
`HouseNumber`	Primary (house) number	*500*
`PreDir`	Street pre-direction	-
`Street`	Street name or PO Box	*108th Ave*
`PostDir`	Street post-direction	*NE*
`Unit`	Secondary (unit) number	*Suite 2200*
`City`	City name or abbreviation	*Bellevue*
`State`	State name or abbreviation	*WA*
`Zip`	5-digit zip code	*98004*
`Zip4`	Last 4-digit zip code	*0000*

2. RELATED WORK

Record linkage is the task of finding records that refer to the same entity across different data sources. Its process is trivial, where the records that relate to the same entity or event all share a common, unique key or identifier. However, there is often no unique key that is shared by all the data collections. The techniques of record linkage mainly rely on an element-wise comparison between pairs of records, each comprising an ensemble of partially identifying personal attributes. These attributes commonly include residential addresses, and standardizing address data is therefore an important step in data pre-processing [5, 1].

[1]Based on the data from *Intelius.com*, as of June, 2013.

LocWeb'14, November 3, 2014, Shanghai, China.

ACM 978-1-4503-1459-6/14/11 $15.00.
http://dx.doi.org/10.1145/2663713.2664430.

In these settings, the techniques of address segmentation can be broadly divided into two groups: deterministic or rule-based techniques, and probabilistic techniques. The most common approach is the manual specification of parsing and transformation rules. A well-known example of this approach is AutoStan [6] in the biomedical research field. [8] also implemented a rule-based system to parse addresses through regular expression matching. Probabilistic methods are an alternative to these rule-based approaches. Statistical models, HMM in particular, have been used extensively to help solve sequence labeling problems, such as part-of-speech tagging [7]. More recently, HMM has been applied to the problem of extracting structured information from unstructured text [3, 4] and address standardization [2].

3. DATA

3.1 Synthetic Training Data

Our address data sources are at a very large scale, such as in billions, and vary in data quality. To experiment with parsing address at such a large scale, we leverage some of the current *Big Data* processing tools, such as *Hadoop*. We apply *Hadoop* to mine nearly 100 million unique addresses from one of our data sources, which is of higher quality than other data sources. These addresses have been pre-segmented into the above fields defined in Table 1 by our data vender, so these extracted addresses can naturally serve as our prototype training data for the HMM-based system. Here is a sample instance:

<HouseNumber>500</HouseNumber>
<Street>108th Ave</Street>
<PostDir>NE</PostDir> <City>Bellevue</City>
<State>WA</State> <Zip>98004</Zip>

But one problem here is that all these segmented instances have already been normalized, such as "Street", "Str.", "St." and other forms of word "street" are normalized into the only form "St". While for many regular words there is only one correct spelling, there are often different written forms for proper names (commonly used as street or locality). Hence, this data source actually lacks normal address variations. Furthermore, these addresses are also well-formed, correctly ordered and complete, so that it does not contain any case that consists of incomplete or badly ordered fields. However, correct addresses may appear in different orders or even miss some fields. The HMM is quite robust with respect to the training set used, so it is quite feasible to add training instances that are archetypes of unusual address patterns, without compromising the performance on more typical addresses. Therefore, we introduce some techniques to automatically generate better synthetic training data from this high-quality data source, which is more close to the real-world scenarios and further improve the system performance. The following approaches are sequentially applied.

1. **Denormalization** We take one step backward to capture all various address expressions through denormalization. We collected four dictionaries of grouped geographical variations. Then if an expression e, which is included in some group g within these dictionaries, appears in a prototype training instance, we randomly select one expression e' from the group g to replace e in that training instance. This procedure is applied to each expression in each prototype training instance. Since e could be different from e', we successfully denormalize the original data source to create a synthetic training data set with more variations.

   (1) *State Dictionary* consists of common variations and abbreviations of a total of 59 state and region names in United States. Such as, all "Florida", "Fla.", and "FL" refer to the "Florida" state;

   (2) *Street Dictionary* lists 195 groups of expressions and various abbreviations about "street", such as "boulevard" can normally be written as "blvd", "boul", "boulv", and "boulevard". Moreover, "post office box" can usually be abbreviated as "po box", "pbox", "pbx", "p o box", "pobx", "pobox", and "pob"[2];

   (3) *Unit Dictionary* contains 20 groups of words and abbreviations of unit expressions, such as both "apt" and "apmt" are familiar forms of "apartment";

   (4) *Direction Dictionary* includes common direction phrases, i.e., both "ne" and "northeast" express the "northeast" direction.

   After applying this *Denormalization* step, the above sample prototype training instance can be transformed to the following example:
   <HouseNumber>500</HouseNumber>
   <Street>108th Avenue</Street>
   <PostDir>Northeast</PostDir>
   <City>Bellevue</City>
   <State>WA</State> <Zip>98004</Zip>

2. **Incompletion** Since real-world addresses may be incomplete while all addresses from this high-quality data source are comprised of complete information, we need some incomplete addresses in our training data to improve the HMM system with the capacity for correctly parsing incomplete addresses. Accordingly, we randomly select 20% from the original prototype training instances. For each field in these instances, we remove it with a 0.5 probability. For example, the above denormalized training instance can be modified as a synthetic training instance with incomplete fields like:
   <Street>108th Avenue</Street>
   <City>Bellevue</City> <State>WA</State>
   <Zip>98004</Zip>

3. **Disarrangment** Reordering the address fields is also necessary to simulate real-world addresses with unusual address field orders. When a person looks for an address in a search engine or provides addresses to others, for instance, the order is not always perfect. A possible reason is that people usually first provide the address fields that they are more confident with, and then try to add more detailed information. But the correctness and accuracy of the information added afterwards may not be guaranteed. Consequently, we randomly rearrange the field order in the newly generated training instances from the above *Incompletion* step, and these rearranged instances are added into the final training data set. One reordered training instance produced from the above incomplete instance is like:

[2]We deliberately put "Post Office Box" information in the `Street` field in this task.

<State>WA</State> <Zip>98004</Zip>
<City>Bellevue</City>
<Street>108th Avenue</Street>

We have experimented with different percentages of prototype training instances using in *Incompletion* and *Disarrangement* steps on a development data set. We find out 20% is appropriate. Otherwise, it may be too small to capture the noisy nature of real-world data, or the HMM may get biased by the synthetic training data and have inferior performance on handling the well-formed address cases.

3.2 Annotation Data

To evaluate the system performance, we construct a small but representative annotated corpus. Two annotators are trained to verify the validity of each address and segment the address into predefined fields if the address is considered as correctly formatted. Otherwise, the addresses would be ignored. The final annotated corpus consists of 587 typical, representative addresses (such as, city address, rural address, highway address, Puerto Rico address, military address, PO Box address, and etc.), which are randomly selected from another raw address data set (different from the above high-quality data set). We reach an inter-annotator agreement of $K = 0.742$ in Kappa. We consider the agreement quite good, considering the number of categories and the difficulties of the task. We find out two annotators usually disagree about the boundaries of `Street` fields, due to that no additional world knowledge is given, such as "*1895 GAMAY TER TE CHULA VISTA CA 91913*". This also reflects the difficulties of this task, even for people.

4. SYSTEM DESCRIPTIONS

4.1 Baseline System

Our rule-based baseline system [8] is designed based on the open-source project *JGeocoder* (2008)[3], which was further developed and customized based on the different cases identified over the course of development. The input raw address string is segmented into the predefined address fields using a library of regular expressions and rule sets. This system has already been used in our company for production purpose, which gives reasonable performance.

4.2 Hidden Markov Model System

We apply some tools provided in the Java Extraction Toolkits (Jet) package[4] to build this system. We treat the tokenized input address as an ordered sequence of observation symbols, and assume that each observation symbol has been emitted by one of the hidden states, such as `Street` and `City`. Within each of the class states, a statistical bigram model is employed, with the standard one-word-per-state emission. Since these probabilities are estimated based on the observations seen in a corpus, back-off models are used to reflect the strength of support for a given statistic. Our HMM system can compute the margin - the difference between the log probabilities of the top two hypotheses. This is used as a rough measure of confidence in the top hypothesis, and a large margin indicates greater confidence in the first hypothesis. Furthermore, our HMM system can also produce the top N best hypotheses for each address.

[3]http://jgeocoder.sourceforge.net/index.html
[4]http://cs.nyu.edu/grishman/jet/license.html

As we know, the addresses provided by people may contain new or misspelled words, which do not appear in the training data. If we apply the HMM system directly to these unknown words, the system may not perform accurately. This usually introduces errors to address segmentation. Hence, we apply a very simple strategy to solve this problem. For each token t in an address, we check whether t is a word (only letters after the removal of punctuations) that appears in the training data. If it is unknown, we replace it with a special token, "UnknownUnknown". Since unknown words usually appear in `Street`, `City`, and `State` fields, we assign this special token a frequency 1 under these three hidden states respectively, so that this unknown token receives a very small emission probability. Finally, we apply the HMM system on the input address and then replace this special token back with the original token in the result. Although this method is very simple, from the experiments, we observe that it can produce correct segmentations with unknown words effectively, such as, the misspelled word "Broadwya" can still be correctly segmented into `Street` in address "*719 Broadwya Floor 7 New York NY 10003*".

5. EXPERIMENTS

It is very efficient, reliable to train the HMM model. The training takes less than 10 hours to finish training the model with all of the 100 million synthetic training instances. All our experiments are performed on a 8-core 2.33 GHz Intel Xeon Linux server with 32 GB RAM.

5.1 System Performance

Addresses are judged to be accurately parsed if all elements of an input address string are placed into the correct fields. Results are also judged on an individual basis for the correctness of each field. For example, the segmentations of address components that are assigned with `Street` tag will also be evaluated individually. Table 2 compares the performance (in *Precision*, *Recall*, and *F-measure* scores) between the baseline system and the HMM-based system, which are evaluated based on the adjudicated annotation data. The results in Table 2 demonstrate that our HMM-based system can generate more reliable, accurate parsed results compared to our baseline system, either in overall or single field performance. Especially for those complex, unusual addresses, the results prove the substantial advantages of our HMM-based system in terms of the parsing accuracy. Take an uncommon Puerto Rico address for example, "*QQ1 Calle Julio Ruedas San Juan PR 00926*", our HMM system can perfectly segment it into correct fields, but the baseline system even fails to recognize it as a validly formatted address.

5.2 Training Data Effects

We also investigate the benefits of the techniques applied to generate the synthetic training data. In Figure 1, we compare the system performance (in F-measure scores) on the original ("OG") high-quality data source, the only denormalized ("DN") training data, and the final training data that has been processed through all *Denormalization* ("DN"), *Incompletion* ("IC"), and *Disarragement* ("DA") steps. Compared to the "OG" performance, it clearly shows that *Denormalization* approach can improve the system performance on almost all individual fields, especially `Street`, `Unit`, and the overall segmented address with around 5% absolute improvement in F-measure. Once we additionally apply *Incomple-*

Table 2: Results Comparison between Baseline System and HMM System

Field		Performance	
		Baseline	HMM
HouseNumber	Precision	1.000	1.000
	Recall	0.991	1.000
	Fmeasure	0.996	1.000
PreDir	Precision	N/A	1.000
	Recall	0.000	1.000
	Fmeasure	N/A	1.000
Street	Precision	0.579	0.961
	Recall	0.577	0.959
	Fmeasure	0.578	0.960
PostDir	Precision	N/A	0.975
	Recall	0.000	1.000
	Fmeasure	N/A	0.987
Unit	Precision	0.882	0.981
	Recall	0.813	0.916
	Fmeasure	0.846	0.947
City	Precision	0.912	0.997
	Recall	0.889	0.985
	Fmeasure	0.901	0.991
State	Precision	1.000	1.000
	Recall	0.939	1.000
	Fmeasure	0.968	1.000
Zip	Precision	0.550	1.000
	Recall	0.546	1.000
	Fmeasure	0.548	1.000
Zip4	Precision	N/A	1.000
	Recall	0.000	1.000
	Fmeasure	N/A	1.000
Address	Precision	0.284	0.956
	Recall	0.284	0.956
	Fmeasure	0.284	**0.956**

tion and *Disarragment* approaches, the performance on each field gets improved more. Some fields, such as `Street` and `PostDir`, receive almost 10% absolute improvement compared to the "DN" scores. The enhancement on `Unit` even reaches more than 30%, which also helps boost the overall performance from 84.0% to 95.6%. The improvements on these fields are also quite reasonable, because these approaches do help better identifying the boundaries of the fields, especially among `Street`, `PostDir`, and `Unit` fields. Figure 1 also demonstrates the efficiency and necessity of integration of these techniques to create the better training data for the HMM-based system.

Given the raw address "*3117 Flowers RD S M Atlanta GA 30341*" as an example, our HMM system trained on the original data source produces the following incorrect result:

<HouseNumber>3117</HouseNumber>
<Street>Flowers RD S M</Street>
<City>Atlanta</City> <State>GA</State>
<Zip>30341</Zip>

However, our system trained on the final synthetic training data can return the following completely correct result:

<HouseNumber>3117</HouseNumber>
<Street>Flowers RD</Street> <PostDir>S</PostDir>
<Unit>M</Unit> <City>Atlanta</City>
<State>GA</State> <Zip>30341</Zip>

5.3 Discussions

We also measure the time that the HMM-based system needs to parse a raw input address, and it turns out that the average processing time is only around 5 milliseconds per address. Distributed architectures, such as *MapReduce* and *Hadoop*, can provide even faster processing. For instance, at this rate, we have scaled the system up to parse billions of real-world addresses on an 80-node *Hadoop* cluster in a couple of hours. This processing time impressively validates the efficiency and scalability of this HMM-based address parsing system in large-scale record linkage problems.

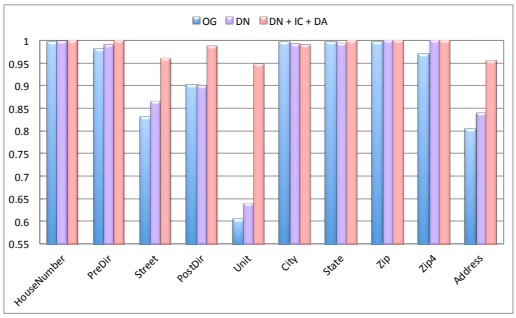

Figure 1: Performance Comparison using Different Techniques for Synthetic Training Data

6. CONCLUSION

Accurately parsing address is a fundamental step in data pre-processing. In this paper, we present an automated probabilistic approach based on HMM, and also introduce several approaches to train models for noisy data. Initial experiments demonstrate that our HMM-based system can correctly parse even complex and unusual addresses with efficiency and capacity for large-scale data sets. In the future, we will investigate more sophisticated machine learning models on processing addresses in such large-scale data settings. Another direction is to explore whether comparable results can be achieved on noisy data through other methods, such as Expectation-Maximization.

7. REFERENCES

[1] M. Cayo and T. Talbot. Positional error in automated geocoding of residential addresses. *International Journal of Health Geographics*, 2003.

[2] P. Christen and D. Belacic. Automated probabilistic address standardisation and verification. In *Australasian Data Mining Conference*, 2005.

[3] D. Freitag and A. McCallum. Information extraction using hmms and shrinkage. In *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[4] D. Freitag and A. McCallum. Information extraction with hmm structures learned by stochastic optimisation. In *Proceedings of the Eighteenth Conference on Artificial Intelligence*, 2000.

[5] J. Han and M. Kamber. Data mining: Concepts and techniques. In *Morgan Kaufmann*, 2000.

[6] MatchWare-Technologies. Autostan and automatch user's manuals. 1998.

[7] L. Rabiner and B.-H. Juang. Fundamentals of speech recognition. *New Jersey, Prentice-Hall, Ch 6*, 1993.

[8] S. Xu, S. Flexner, and V. Carvalho. Geocoding billions of addresses: Toward a spatial record linkage system with big data. In *GiBDA'12: Workshop on GIScience in the Big Data Age*, 2012.