Abstract

In this paper, we address the (to the best of our knowledge) new problem of extracting a structured description of real estate properties from their natural language descriptions in classifieds. We survey and present several models to (a) identify important entities of a property (e.g., rooms) from classifieds and (b) structure them into a tree format, with the entities as nodes and edges representing a part-of relation. Experiments show that a graph-based system deriving the tree from an initially fully connected entity graph, outperforms a transition-based system starting from only the entity nodes, since it better reconstructs the tree.

1 Introduction

In the real estate domain, user-generated free text descriptions form a highly useful but unstructured representation of real estate properties. However, there is an increasing need for people to find useful (structured) information from large sets of such descriptions, and for companies to propose sales/rentals that best fit the clients’ needs, while keeping human reading effort limited. For example, real estate descriptions in natural language may not be directly suited for specific search filters that potential buyers want to apply. On the other hand, a hierarchical data structure representing the real estate property enables specialized filtering (e.g., based on the number of bedrooms, number of floors, or the requirement of having a bathroom with a toilet on the first floor), and is expected to also benefit related applications such as automated price prediction (Pace et al., 2000; Nagaraja et al., 2011).

Our primary objective is to define the new real estate structure extraction problem, and explore its solution using combinations of state-of-the-art methods, thus establishing its difficulty by obtaining performance results for future reference. More specifically, we contribute with: (i) the definition of the real estate extraction problem, amounting to a tree-like structured representation of the property (the property tree) based on its natural language description; (ii) the introduction of structured learning methods that solve the newly defined problem; and (iii) experimental evaluation of the systems on a newly created and annotated real-world data set. For part (ii), we break down the problem into simpler components, using (1) Conditional Random Fields (CRFs) for real estate entity recognition (where entities are floors, rooms, sub-spaces in rooms, etc.), (2) non-projective dependency parsing to predict the part-of relationships between such entities (comparing local and global graph-based, and transition-based algorithms), and (3) a maximum spanning tree algorithm for decoding the desired property tree.

2 Related work

The challenge in structured prediction largely stems from the size of the output space. Specifically in NLP, for sequence labeling (e.g., named entity recognition), which is the first building block of our system, a number of different methods have been proposed, namely CRFs (Lafferty et al., 2001), Maximum Margin Markov Network (M^3N) (Taskar et al., 2003), SVM^struct (Tsochantaridis et al., 2004) and SEARN (Daumé III et al., 2009).

We exploit dependency parsing methods for the construction of the property tree which is similar to the problem of learning the dependency arcs of a sentence. Dependency parsing research has focused on both graph-based and transition-based
parsers. McDonald et al. (2005; 2007) have shown that treating dependency parsing as the search of the highest scoring maximum spanning tree in graphs yields efficient algorithms for both projective (dependencies are not allowed to cross) and non-projective (crossing dependencies are allowed) trees. Later, Koo et al. (2007), adapted the Matrix-Tree Theorem (Tutte, 2001) for globally normalized training over all non-projective dependency trees. On the other hand, transition-based dependency parsing aims to predict a transition sequence from an initial to some terminal configuration and handles both projective and non-projective dependencies (Nivre, 2003; Nivre, 2009). Recent advances on those systems involve neural scoring functions (Chen and Manning, 2014) and globally normalized models (Andor et al., 2016).

More recently, a substantial amount of work (Kate and Mooney (2010), Li and Ji (2014), Miwa and Sasaki (2014) and Li et al. (2016)) jointly considered the two subtasks of entity recognition and dependency parsing. Our work is different since we aim to handle directed spanning trees, or equivalently non-projective dependency structures (i.e., the entities involved in a relation are not necessarily adjacent in the text since other entities may be mentioned in between), which complicates parsing.

3 Structured prediction of real estate properties

We now present the real estate extraction problem and our proposed proof-of-concept solutions.

3.1 Problem formulation

We define entities and entity types for our real estate extraction task. We define an entity as an unambiguous, unique part of a property with independent existence (e.g., bedroom, kitchen, attic). We define as entity mention, a textual phrase (e.g., “a small bedroom”) that we can potentially link to one or more of the entities and whose semantic meaning unambiguously represents a specific entity. Each entity can occur several times in the text, possibly with different mentions and we further classify entities into types as listed in Table 1.

The goal of our structured prediction task is to convert the given input text to a structured representation in the form of a so-called property tree, as illustrated in Fig. 1. That conversion implies both the detection of entities of various types (the “house” property entity, and the spaces “living room”, “kitchen”, etc.) as well as the part-of dependencies between them (e.g., that the “kitchen” is a part of the “house”). We cast the tree construction given the entities as a dependency parsing task over the search of the most probable property tree, since (i) this means decisions on all possible part-of relations are taken jointly (e.g., a certain room can only be part of a single floor), and (ii) we can deal with the fact that there are no hard a priori constraints on the types of entities that can be part of others (e.g., a room can be either part of a floor, or the property itself, like an apartment). It’s worth mentioning that dependency annotations for our problem exhibit a significant number of non-projective arcs (26%), meaning that entities involved in the part-of relation are non-adjacent (i.e., interleaved by other entities), as intuitively expected.

3.2 Structured prediction model

We now describe the constituents of our pipeline to solve the property tree extraction from natural language ads, as sketched in Fig. 2: (1) recognize the entity mentions (Section 3.2.1), then (2) identify the part-of dependencies between those entity mentions (Section 3.2.2), and finally (3) construct the tree structure of the property (e.g., as in Fig. 1). In step (2), we focus on comparing lo-
cally and globally trained graph-based models and
a transition-based one. We only explicitly perform
step (3) in graph-based models, by applying the
maximum spanning-tree algorithm (Chu and Liu,
1965; Edmonds, 1967) for the directed case (see
McDonald et al. (2005)). As an alternative, we
use a transition-based system, which by definition
deals with non-projective trees, and does not need
spanning tree inference.

3.2.1 Sequence labeling
The first step in our structured prediction baseline
is a sequence labeling task, similar to NER: given
a real estate ad’s plain text, we extract the entity
mention boundaries and map the type of the entity
mentions. We adopt linear chain CRFs, a special
case of the CRF algorithm (Lafferty et al., 2001;
Peng and McCallum, 2006), widely used for the
problem of sequence labeling.

3.2.2 Part-of tree construction
The aim of this component is to connect each en-
tity to its parent. This is similar to dependency
parsing but instead of mapping the whole sen-
tence, we map only the identified entity set x (e.g.,
house) to a dependency structure y. Given the en-
tity set x with n terms, a dependency is a tuple
(p, c) where p ∈ {0, ..., n} is the index of the par-
ent term in entity set x, p = 0 is the root-symbol
(only appears as parent) and c ∈ {1, ..., n} is the
index of the child term in the entity set. We use
D(x) to refer to all possible dependencies of x and
T(x) to all possible dependency structures.

We now present our approaches to solve this
part-of tree construction problem.

Locally trained model (Threshold/Edmonds)
We focus on local discriminative training meth-
ods (Yamada and Matsumoto, 2003) where a bi-
ary classifier learns the part-of relation model
(step (2)). Given a candidate parent-child pair,
the classifier scores reflect how likely the part-of
relation holds. The output is then used for the
next and final step (3) of constructing the prop-
ergy tree. Specifically, we construct a fully con-
nected directed graph G = (V, E) with the en-
tities as nodes V, and edges E representing the
part-of relation with the respective classifier scores
as weights. A naive approach to obtain the tree
prediction is threshold-based: keep all edges with
weights exceeding a threshold. This is obviously
not guaranteed to end up being a tree and might
even contain cycles. Our approach directly aims
at finding the maximum spanning tree inside the
(directed) graph to enforce a tree structure. To this
end, techniques designed for dependency parsing
in natural text can be used, more in particular we
use Edmonds’ algorithm (McDonald et al., 2005).

Globally trained model (MTT)
The Matrix-Tree theorem (MTT) (Koo et al.,
2007) provides the algorithmic framework to train
globally normalized models that involve directed
spanning trees, i.e., score parse trees for a given
sentence. Assume we have a vector θ in which
each value θh,m ∈ ℝ corresponds to a weight
∀(h, m) ∈ D(x). The conditional distribution
over all dependency structures y ∈ T(x) is:

\[ P(y|x; \theta) = \frac{1}{Z(x; \theta)} \exp \left( \sum_{h,m \in y} \theta_{h,m} \right) \]

normalized by the partition function Z(x; θ),
which would require a summation over the expo-
entially large number of all possible dependency
structures in T(x). However, the MTT allows di-
rectly computing Z(x; θ) as det(L(θ)), in which
L(θ) is the Laplacian matrix of the graph.

Transition-based dependency parsing (TB)
Given that our system needs to be able to han-
dle non-projective dependency arcs, we employ a
greedy transition-based parsing system (Nivre,
2009; Bohnet and Nivre, 2012) as the basis of
our parser. The system is defined as a configu-
ration C = (Σ, B, A) which consists of Σ the
stack, B the buffer and A the set of dependency
archs. The aim is, given an initial configuration
and a set of permissible actions, to predict a tran-
sition sequence to some terminal configuration to
derive a dependency parse tree. We define the ini-
tial configuration for an entity set x = w_1, ..., w_n
to be ([root],[w_1, ..., w_n],{}) and the terminal con-
figuration ([0],[],A) (for any arc set A). The first
three actions (LEFT-ARC, RIGHT-ARC, SHIFT)
are defined similar to arc-standard systems (Nivre,
Table 2: Performance of the real estate entity recognition with hyperparameter $\lambda_{\text{CRF}} = 10$.

<table>
<thead>
<tr>
<th>Entity type</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>3170</td>
<td>1912</td>
<td>2217</td>
<td>0.62</td>
<td>0.59</td>
<td>0.61</td>
</tr>
<tr>
<td>floor</td>
<td>2685</td>
<td>515</td>
<td>529</td>
<td>0.84</td>
<td>0.84</td>
<td>0.84</td>
</tr>
<tr>
<td>space</td>
<td>11952</td>
<td>2053</td>
<td>2003</td>
<td>0.85</td>
<td>0.86</td>
<td>0.86</td>
</tr>
<tr>
<td>subspace</td>
<td>4338</td>
<td>575</td>
<td>1181</td>
<td>0.88</td>
<td>0.79</td>
<td>0.83</td>
</tr>
<tr>
<td>field</td>
<td>2083</td>
<td>700</td>
<td>718</td>
<td>0.75</td>
<td>0.74</td>
<td>0.75</td>
</tr>
<tr>
<td>extra building</td>
<td>253</td>
<td>34</td>
<td>143</td>
<td>0.88</td>
<td>0.64</td>
<td>0.74</td>
</tr>
<tr>
<td>Overall</td>
<td>24481</td>
<td>5789</td>
<td>6791</td>
<td>0.81</td>
<td>0.78</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Table 3: Performance of the three approaches on the structured prediction task. The top half are results for known entities (i.e., the gold standard as annotated), while the bottom half starts from the entities as found in step (1) of our end-to-end pipeline ($\lambda_{\text{CRF}} = 10$ and $C = 1$).

<table>
<thead>
<tr>
<th>Model</th>
<th>TP</th>
<th>FP</th>
<th>FN</th>
<th>Precision</th>
<th>Recall</th>
<th>$F_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>known entities</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thresh.</td>
<td>15723</td>
<td>6365</td>
<td>16461</td>
<td>0.71</td>
<td>0.49</td>
<td>0.58</td>
</tr>
<tr>
<td>Edm.</td>
<td>22058</td>
<td>10126</td>
<td>10126</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
</tr>
<tr>
<td>MTT</td>
<td>22361</td>
<td>9823</td>
<td>9823</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
</tr>
<tr>
<td>TB</td>
<td>14816</td>
<td>17368</td>
<td>17368</td>
<td>0.46</td>
<td>0.46</td>
<td>0.46</td>
</tr>
<tr>
<td>full pipeline</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thresh.</td>
<td>9309</td>
<td>9846</td>
<td>22965</td>
<td>0.49</td>
<td>0.29</td>
<td>0.36</td>
</tr>
<tr>
<td>Edm.</td>
<td>12859</td>
<td>17417</td>
<td>19415</td>
<td>0.42</td>
<td>0.40</td>
<td>0.41</td>
</tr>
<tr>
<td>MTT</td>
<td>12426</td>
<td>17850</td>
<td>19848</td>
<td>0.41</td>
<td>0.39</td>
<td>0.40</td>
</tr>
<tr>
<td>TB</td>
<td>9677</td>
<td>19043</td>
<td>22507</td>
<td>0.34</td>
<td>0.30</td>
<td>0.32</td>
</tr>
</tbody>
</table>

Better separable, as expected, since the mentions do not vary a lot (e.g., “shower”, “bedroom”).

4.3 Dependency parsing

The upper part of Table 3 lists the performance for the dependency parsing subtask by itself, assuming perfect real estate entity recognition: for this evaluation we used the gold standard provided by the annotations. We measure the performance on the threshold-based model, the logistic regression and the MTT scorings followed by Edmonds’ algorithm for directed graphs to enforce a tree structure and the transition-based (TB) model. Note that in the case of known entities we have that there are exactly as many false positives as false negatives, since an incorrect edge prediction (FP) implies that the correct one has not been predicted (FN), and vice versa, because of the enforced tree structure that has to cover all entities. As expected, the MTT approach performs better than the others, because the globally trained model learns directed spanning trees. Predicting the maximum spanning tree (Edmonds’) achieves higher $F_1$ score than simply considering the predictions of the classifier without any structural enforcement (threshold-based). The TB class of parsers is of great interest because of their speed, state-of-the-art performance (Andor et al., 2016) and the potential to be extended towards joint models (future work), although in our comparative study they tend to perform slightly worse than the graph-based parsers, because of subsequent error propagation (Chen and Manning, 2014).

---

1 https://www.realo.be/en
2 https://github.com/bekou/ad_data
4.4 Pipeline approach

The bottom rows in Table 3 refer to the pipeline approach combining both sequence labeling and dependency parsing subtasks: input entities for the parser are not necessarily correct. Given a new real estate ad, first the CRF identifies the entity mention token boundaries and then the tree structure among the extracted entities is constructed. The locally trained approach yields marginally better performance than MTT: MTT learns spanning tree sequences as a whole, so it is harder to connect segments that are incorrect or incomplete. The TB system exhibits the same performance as in the case where entities were known, but we think that incorporating neural scoring functions (Chen and Manning, 2014) or using beam-search instead of using the greedy approach will improve performance (Andor et al., 2016).

5 Conclusion

In this paper, we presented a comparative study on the newly defined problem of extracting the structured description of real estate properties. We divided the problem into the sub-problems of sequence labeling and non-projective dependency parsing since existing joint models are restricted to non-crossing dependencies. Overall, MTT outperforms other approaches when the entities are known while adopting a maximum spanning tree algorithm using individual scored edge weights seems to be marginally better in our pipeline.

Acknowledgments

The presented research was performed within the MALIBU project, funded by Flanders Innovation & Entrepreneurship (VLAIO).

References


