Reconstructing the house from the ad: Structured prediction on real estate classifieds

Giannis Bekoulis, Johannes Deleu, Thomas Demeester, Chris Develder
Ghent University – imec
Belgium
{firstname.lastname}@ugent.be

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

Entity type	Description	Examples
property	The property.	bungalow, apartment
floor	A floor in a building.	ground floor
space	A room within the building.	bedroom, bathroom
subspace	A part of a room.	shower, toilet
field	An open space inside or outside the building.	bbq, garden
extra building	An additional building which is also part of the property.	garden house

Table 1: Real estate entity types.

Original ad:
The property includes an apartment house with a garage. The house has living room, kitchen and bathroom with shower.

Structured representation:
```
house       | mention='apartment house'
  living room   | mention='living room'
  kitchen       | mention='kitchen'
  bathroom      | mention='bathroom'
    shower      | mention='shower'
garage          | mention='garage'
```

Figure 1: Sample unstructured ad and corresponding structured representation as a property tree.

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-

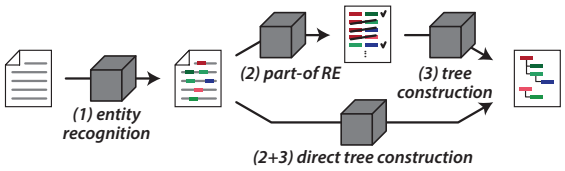

Figure 2: The full structured prediction pipeline.

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 entity to its parent. This is similar to dependency parsing but instead of mapping the whole sentence, we map only the identified entity set x (e.g., house) to a dependency structure y. Given the entity set x with n terms, a dependency is a tuple (p, c) where $p \in \{0, ..., n\}$ is the index of the parent term in entity set x, $p = 0$ is the root-symbol (only appears as parent) and $c \in \{1, ..., n\}$ 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 methods (Yamada and Matsumoto, 2003) where a binary 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 *property tree*. Specifically, we construct a fully connected directed graph $G = \{V, E\}$ with the entities 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 $\theta_{h,m} \in \mathbb{R}$ corresponds to a weight $\forall(h, m) \in D(x)$. The conditional distribution over all dependency structures $y \in T(x)$ is:

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

normalized by the partition function $Z(x;\theta)$, which would require a summation over the exponentially large number of all possible dependency structures in $T(x)$. However, the MTT allows directly computing $Z(x;\theta)$ as $\det(L(\theta))$, in which $L(\theta)$ is the Laplacian matrix of the graph.

Transition-based dependency parsing (TB)

Given that our system needs to be able to handle 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 configuration $C = (\Sigma, B, A)$ which consists of Σ the stack, B the buffer and A the set of dependency arcs. The aim is, given an initial configuration and a set of permissible actions, to predict a transition sequence to some terminal configuration to derive a dependency parse tree. We define the initial configuration for an entity set $x = w_1, ..., w_n$ to be ([root],[$w_1, ..., w_n$],{}) and the terminal configuration ([0],[],A) (for any arc set A). The first three actions (LEFT-ARC, RIGHT-ARC, SHIFT) are defined similar to arc-standard systems (Nivre,

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

Table 2: Performance of the real estate entity recognition with hyperparameter $\lambda_{\text{CRF}} = 10$.

2003) for projective dependency parsing. In addition, the SWAP operation reorders the input words, thus allowing to derive non-projective trees (Nivre, 2009).

4 Experimental results

We present results for the total real estate framework as well as for each step individually.

4.1 Experimental setup

We collected 887,599 Dutch property advertisements from a real estate company.[1] Three human annotators manually annotated 2,318 ads (1 annotation per ad, $\sim$773 ads per annotator) by creating the property tree of the advertisements. The dataset is available for research purposes, see our github codebase.[2] In our experiments, we use only the annotated text advertisements. We implemented the local model, the MTT and the non-projective transition-based system. The code thereof is available on github.[2] We also use our own CRF implementation. We measure precision, recall, and F_1 on the test set, and report averaged values in a 5-fold cross-validation setting.

4.2 Entity extraction

Table 2 presents our results for the sequence labeling subtask. We separately show the performance of our model for each entity type (see Table 1). Overall, the CRF performs well with a score of $F_1 = 0.80$. Specifically, space is the best performing entity type. Note that the space entity type is the most frequent one in our table. On the other hand, property is the least represented class, since the ads usually mention the property type only once. The performance of the property class is lower because it can have a wide range of values (e.g., "helios apartments", "milos villa"). Moreover, the entity mentions for the space type are better separable, as expected, since the mentions do not vary a lot (e.g., "shower", "bedroom").

[1] https://www.realo.be/en

[2] https://github.com/bekou/ad_data

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

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 $C = 1$).

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).

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

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August. Association for Computational Linguistics.

Bernd Bohnet and Joakim Nivre. 2012. A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1455–1465, Jeju Island, Korea, July. Association for Computational Linguistics.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, Doha, Qatar, October. Association for Computational Linguistics.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14:1396–1400.

Hal Daumé III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning Journal (MLJ)*, 75(3):297–325, June.

Jack Edmonds. 1967. Optimum branchings. *Journal of research of the National Bureau of Standards*, 71B(4):233–240.

Rohit J. Kate and Raymond Mooney. 2010. Joint entity and relation extraction using card-pyramid parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 203–212, Uppsala, Sweden, July. Association for Computational Linguistics.

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. 2007. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, Prague, Czech Republic, June. Association for Computational Linguistics.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001)*, pages 282–289, Massachusetts, USA, July. Morgan Kaufmann.

Qi Li and Heng Ji. 2014. Incremental joint extraction of entity mentions and relations. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 402–412, Baltimore, Maryland, June. Association for Computational Linguistics.

Fei Li, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Joint models for extracting adverse drug events from biomedical text. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI 2016)*, pages 2838–2844, New York, USA, July. IJCAI/AAAI Press.

Ryan McDonald and Fernando Pereira. 2007. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 81–88, Trento, Italy, April. Association for Computational Linguistics.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 523–530, Vancouver, British Columbia, Canada, October. Association for Computational Linguistics.

Makoto Miwa and Yutaka Sasaki. 2014. Modeling joint entity and relation extraction with table representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1858–1869, Doha, Qatar, October. Association for Computational Linguistics.

Chaitra H. Nagaraja, Lawrence D. Brown, and Linda H. Zhao. 2011. An autoregressive approach to house price modeling. *The Annals of Applied Statistics*, 5(1):124–149, March.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 149–160, Nancy, France, April.

Joakim Nivre. 2009. Non-projective dependency parsing in expected linear time. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 351–359, Suntec, Singapore, August. Association for Computational Linguistics.

Kelley Pace, Ronald Barry, Otis W. Gilley, and C.F. Sirmans. 2000. A method for spatialtemporal forecasting with an application to real estate prices. *International Journal of Forecasting*, 16(2):229 – 246, April.

Fuchun Peng and Andrew McCallum. 2006. Information extraction from research papers using conditional random fields. *Information processing & management*, 42(4):963–979, July.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin markov networks. In *Advances in neural information processing systems*, volume 16, pages 25–32. MIT Press.

Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. 2004. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-first International Conference on Machine Learning*, page 104, Alberta, Canada, July. ACM.

William T. Tutte. 2001. Graph theory. In *Encyclopedia of Mathematics and its Applications*, volume 21, page 138. Cambridge University Press.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pages 195–206, Nancy, France, April.