Application-Driven Relation Extraction with Limited Distant Supervision

Andreas Vlachos
Computer Science Department
University College London
a.vlachos@cs.ucl.ac.uk

Stephen Clark
Computer Laboratory
University of Cambridge
sc609@cam.ac.uk

Abstract

Recent approaches to relation extraction following the distant supervision paradigm have focused on exploiting large knowledge bases, from which they extract substantial amount of supervision. However, for many relations in real-world applications, there are few instances available to seed the relation extraction process, and appropriate named entity recognizers which are necessary for pre-processing do not exist. To overcome this issue, we learn entity filters jointly with relation extraction using imitation learning. We evaluate our approach on architect names and building completion years, using only around 30 seed instances for each relation and show that the jointly learned entity filters improved the performance by 30 and 7 points in average precision.

1 Introduction

In this paper we focus on relation extraction in the context of a real-world application. The application is a dialog-based city tour guide, based in Edinburgh. One of the features of the system is its pro-active nature, offering information which may be of interest to the user. In order to be pro-active in this way, as well as answer users' questions, the system requires a large amount of knowledge about the city. Part of that knowledge is stored in a database, which is time-consuming and difficult to populate manually. Hence, we have explored the use of an automatic knowledge base population technique based on distant supervision (Craven and Kumlien, 1999; Mintz et al., 2009).

The attraction of this approach is that the only input required is a list of seed instances of the relation in question and a corpus of sentences expressing new instances of that relation. However, existing studies typically assume a large seed set, whereas in our application such sets are often not readily available, e.g. Mintz et al. (2009) reported using 7K-140K seed instances per relation as input. In this paper, the two relations that we evaluate on are architect name and completion year of buildings. These were chosen because they are highly relevant to our application, but also somewhat non-standard compared to the existing literature; and crucially they do not come with a readily-available set of seed instances.

Furthermore, previous approaches typically assume named entity recognition (NER) as a pre-processing step in order to construct the training and testing instances. However, since these tools are not tailored to the relations of interest, they introduce spurious entity matches that are harmful to performance as shown by Ling and Weld (2012) and Zhang et al. (2013). These authors ameliorated this issue by learning fine-grained entity recognizers and filters using supervised learning. The labeled data used was extracted from the anchor text of entity mentions annotated in Wikipedia, however this is not possible for entities not annotated in this resource.

In this work, instead of relying on labeled data to construct entity filters, we learn them jointly with the relation extraction component. For this purpose we use the imitation learning algorithm DAGGER (Ross et al., 2011), which can handle the dependencies between actions taken in a sequence, and use supervision for later actions to learn how to take actions earlier in the sequence. We evaluate our approach using around 30 seed instances per relation and show that the jointly learned entity filters result in gains of 7 and 30 points in average precision for the completion year and the architect name relations respectively.

Proceedings of the AHA! Workshop on Information Discovery in Text, pages 1–6,
Dublin, Ireland, August 23 2014.

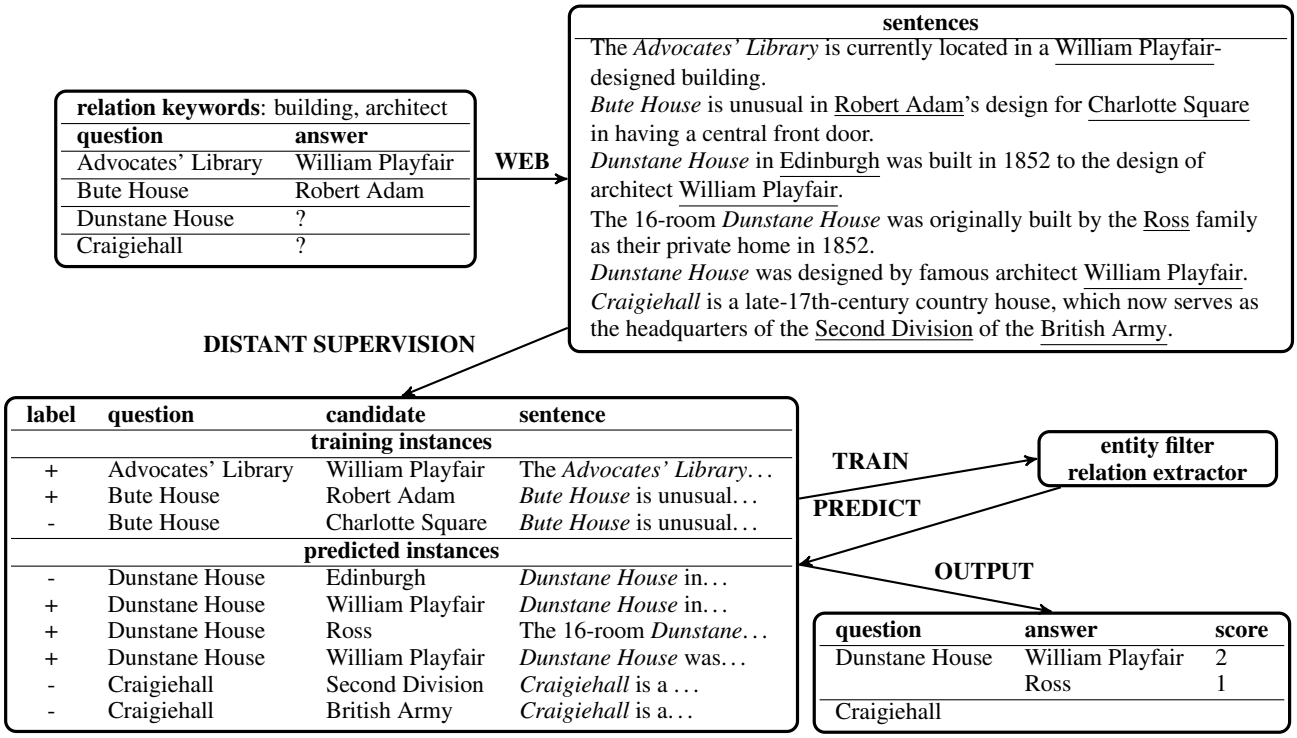

Figure 1: The stages of our proposed approach applied to the architect name relation.

2 Approach overview

We will use the architect-building relation as an example to give an overview of our approach, as shown in Figure 1. The input to the system is a list of buildings, where for some we know the architect (the seeds), and the task is to find the architects for the remainder. One difference with the standard setup for relation extraction using distant supervision is that we assume a list of historical buildings instead of a tailored NER system. This is reasonable for the example, since such a list is relatively easy to acquire. In order to create training data, queries containing words from the seeds are sent to a search engine. Sentences from the returned pages are then processed to find examples which contain mentions of both a building and the corresponding architect. Applying the distant supervision hypothesis, we assume that such sentences are indeed expressing the desired relation, and these are positive examples. While such data contains noise, it has been shown to be useful in practice (Yao et al., 2010; Hoffmann et al., 2011).

At test time the input is the name of a historical building. Now the web is searched to find example sentences containing this name, and the classifier is applied to each sentence, returning either the name of the architect, or none. Note that different sentences could provide evidence for different architects; hence assuming only one architect for each building, a procedure is required to decide between the possible answers (see Sec. 5).

3 Entity Filtering for Relation Extraction

Each relation extraction instance consists of a sentence containing a question entity (e.g. *Bute House*) and a candidate answer (e.g. *Robert Adam*), and the task is to predict whether the answer and question entity have the relation of interest. The standard approach is to learn a binary classifier (possibly as part of a more complex model e.g. Hoffmann et al. (2011)) using features that describe each entity as well as the lexico-syntactic relation between them in the sentence. These commonly include the lexicalized dependency path from the question entity to the candidate answer, as well as the lemmas on this path. In this setup, NER assists by filtering the instances generated to those that contain appropriate recognized entities and by providing features for them.

However, since we do not assume NER in pre-processing, this task becomes harder in our setup, since the candidate answers are very often inappropriate for the relation at question. A simple way

Algorithm 1: Learning with DAGGER

Input: training set $\mathcal{S}$, loss ℓ, CSC learner *CSCL*
Output: Learned policy H_N

1 CSC Examples $E = \emptyset$
2 **for** $i = 1$ **to** N **do**
3 **for** s ***in*** S **do**
4 Predict $\hat{y}_{1:T} = H_{i-1}(s)$
5 **for** $\hat{y}_t$ ***in*** $\pi(s)$ **do**
6 Extract features $\Phi_t = f(s, \hat{y}_{1:t-1})$
7 **foreach** *possible action* y_t^j **do**
8 Predict $y\prime_{t+1:T} = H_{i-1}(s; \hat{y}_{1:t-1}, y_t^j)$
9 Assess $c_t^j = \ell(\hat{y}_{1:t-1}, y_t^j, y\prime_{t+1:T})$
10 Add (Φ_t, c_t) to E
11 Learn $H_i = CSCL(E)$

to incorporate NER-like information is to add the features that would have been used for NER to the relation extraction features and learn a classifier as above. Such features are commonly extracted from the candidate answer itself as well as its context. The former include the tokens of the answer, their lemmas, whether the answer is capitalised, etc. The latter include the words and bigrams preceding and following the answer, as well as syntactic dependencies between the words denoting the entity and surrounding lemmas.

However, while these features are likely to be useful, they also render learning relation extraction harder because they are not directly relevant to the task. For example, the features describing the first training instance of Fig. 1 would include that the token *Playfair* is part of the candidate answer and that the lemma *design* is part on the syntactic dependency path between the architect and the building, but only the latter is crucial for the correct classification of this instance. Thus, including the NER features about the candidate answer can be misleading, especially since they tend to be less sparse than the relation extraction ones.

Therefore we split the prediction into two binary classification stages: the first stage predicts whether the candidate answer is appropriate for the relation (entity filtering), and the second one whether the sentence expresses the relation between the answer and the question entity (relation extraction). If the prediction for the first stage is negative, then the second stage is not reached. However, we do not have labels to train a classifier for the entity filtering stage since if an instance is negative this could be either due to the candidate answer or to the relation expressed in the sentence. We discuss how we overcome this issue using the algorithm DAGGER (Ross et al., 2011) next.

4 Imitation learning with DAGGER

Imitation learning algorithms such as DAGGER and SEARN (Daumé III et al., 2009) have been applied successfully to a variety of structured prediction tasks (Vlachos, 2012; He et al., 2013) due to their flexibility in incorporating features. In this work we focus on the parameter-free version of DAGGER and highlight its ability to handle missing labels in the training data. During training, DAGGER converts the problem of learning how to predict sequences of actions into cost sensitive classification (CSC) learning. The dependencies between the actions are learned by appropriate generation of CSC examples. In our case, each instance is predicted by a sequence of two actions: an entity filtering action followed (if positive) by a relation extraction action. The output is a learned policy, consisting of the binary classifiers for entity filtering and relation extraction.

Following Alg. 1, in each iteration DAGGER generates training examples using the previous learned policy H_{i-1} to predict the instances (line 4). For each action taken, the cost for each possible action is estimated by assuming that the action was taken; then the following actions for that instance are predicted

	Recall-*top*	Precision-*top*	F-score-*top*	Recall-*all*	Precision-*all*	F-score-*all*
Base	0.28	0.28	0.28	0.9	0.1	0.18
1stage	0.52	0.71	0.6	0.67	0.68	0.675
2stage	0.5	0.68	0.58	0.67	0.67	0.67
Base	0.0	0.0	0.0	0.62	0.002	0.004
1stage	0.15	0.26	0.19	0.23	0.17	0.2
2stage	0.26	0.65	0.37	0.3	0.55	0.39

Table 1: Test set results for the 3 systems on year completed (top) and architect name (bottom).

using H_{i-1} (line 8); and the complete sequence of actions is compared against the correct output using the loss function (line 9). Since the latter is only applied to complete sequences, it does not need to decompose over individual actions. We define the loss to be 0 when the relation extraction stage is correct and 1 otherwise. Therefore we do not need to know the labels for entity filtering, but we learn a classifier for it so that the relation extraction predictions are correct. Finally, the CSC training examples generated are added (line 10) and a new policy is learnt (line 11).

Since the losses are either 0 or 1, the CSC learning task is equivalent to ordinary classification learning. To learn the binary classifiers for each stage we implemented the adaptive regularization of weights (AROW) algorithm (Crammer et al., 2009) which scales to large datasets and handles sparse feature sets by adjusting the learning rate for each feature. In the first iteration, we do not have a learned policy, thus we assume a naive entity filter that accepts all candidate answers and a relation extractor that predicts the correct label.

5 Experiments

The relations used for evaluation are building-architect and building-completion_year, for the reasons given in Sec. 1. For each of the 138 listed historical buildings in Wikipedia,[1] we found the correct answers, resulting in 60 building-completion_year and 68 building-architect pairs. We split the data into two equal parts for training/development and testing. We then collected relevant web pages querying the web as described in Sec. 2. The queries were submitted to Bing via its Search API and the top 300 results for each query were obtained. We downloaded the corresponding pages and extracted their textual content with BoilerPipe (Kohlschütter et al., 2010). We then processed the texts using the Stanford CoreNLP toolkit.[2] We tried to match the question entity with tokens in each of the sentences, allowing for minor differences in tokenization, whitespace and capitalization. If a sentence contained the question entity and a candidate answer, we parsed it using the Klein and Manning (2002) parser. The instances generated were labeled using the distant supervision assumption, resulting in 974K and 4.5M labeled instances for the completion year and the architect relation, respectively.

We ran experiments with three systems; the jointly learned entity filtering-relation extraction approach using imitation learning (henceforth 2stage), the one-stage classification approach using the features for both entity filtering and relation extraction (henceforth 1stage), and a baseline that for each question entity returns all candidate answers for the relation ranked by the number of times they appeared with the question entity and ignoring all other information (henceforth Base). Following four-fold cross-validations experiment on the development data, we used 12 iterations for learning with DAGGER.

Each system returns a list of answers ranked according to the number of instances classified as positive for that answer. We used two evaluation modes. The first considers only the top-ranked answer (*top*), whereas the second considers all answers returned until either the correct one is found or they are exhausted (*all*). In *all* we define recall as the number of correct answers over the total number of question entities, and precision as the chance of finding the correct answer while traversing those returned.

Results by all models are reported for both relations in Table 1. A first observation is that the architect name relation is substantially harder to extract since all models achieve worse scores than for the completion year relation. More specifically, Base achieves respectable scores in *top* mode in completion year extraction, but it fails completely in architect name. This is due to the existence of many other names

[1] http://en.wikipedia.org/wiki/Category:Listed_buildings_in_Edinburgh

[2] http://nlp.stanford.edu/software/corenlp.shtml

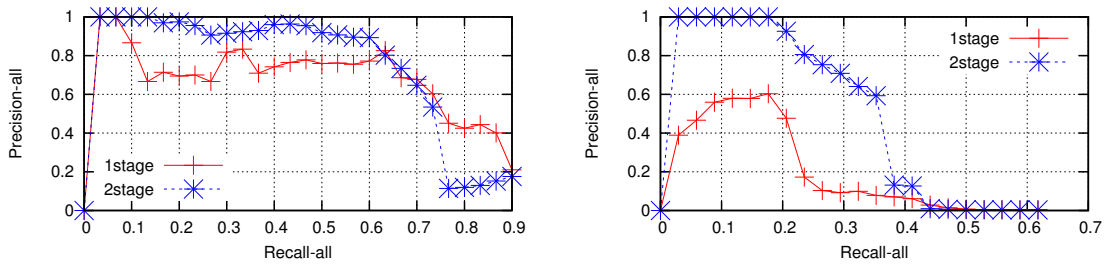

Table 2: Test set precision-recall curves in *all* mode for year completed (left) and architect name (right).

that appear more frequently together with a building than that of its architect, while the completion year is sometimes the number most frequently mentioned in the same sentence with the building. In addition, Base achieves the maximum possible *all* recall by construction, since if there is a sentence containing the correct answer for a question entity it will be returned. However this comes at a cost of low precision.

Both the machine-learned models improve upon Base substantially on both datasets, with the 2stage model being substantially better in architect name extraction, especially in terms of precision. In completion year extraction the differences are smaller, with 1stage being slightly better. These small differences are expected since recognizing completion years is much easier than recognizing architect names, thus learning a separate entity filtering model for them is less likely to be useful. Nevertheless, inspecting the weights learned by the 2stage model showed that some useful distinctions were learned, e.g. being preceded by the word "between" as in "built between 1849 and 1852" renders a number less likely to be a completion year. Finally, we examined the quality of the learned models further by generating precision-recall curves for the *all* mode by adjusting the classification thresholds used by 1stage and 2stage. As shown in the plots of Table 2, 2stage achieves higher precision than 1stage at most recall levels for both relations, with the benefits being more pronounced in the architect name relation. Summarizing these curves using average precision (Manning et al., 2008), the scores were 0.69 and 0.76 for the completion year, and 0.21 and 0.51 for the architect, for the 1stage and the 2stage models respectively, thus confirming the usefulness of separating the entity filtering features from relation extraction.

6 Discussion

While all the buildings considered in our experiments have a dedicated Wikipedia page, only a few had a sentence mentioning them together with the correct answer in that resource. Also, the architects who were the correct answers did not always have a dedicated Wikipedia page. Even though combining a search engine with distant supervision results in a highly imbalanced learning task, it increases the potential coverage of our system. In this process we rely on the keywords used in the queries in order to find pages containing the entities intended rather than synonymous ones, e.g. the keyword "building" helps avoid extracting sentences mentioning saints instead of churches. Nevertheless, building names such as churches named after saints were often ambiguous resulting in false positives.

Bunescu and Mooney (2007) also used a small seed set and a search engine, but they collected sentences via queries containing both the question and the answer entities, thus (unreallistically) assuming knowledge of all the correct answers. Instead we rely on simple heuristics to identify candidate answers. These heuristics are relation-dependent and different types of answers can be easily accommodated, e.g. in completed year relation they are single-token numbers. Finally, the entity filters learned jointly with relation extraction in our approach, while they perform a role similar to NER, they are learned so that they help avoid relation extraction errors and not to replace an actual NER system.

7 Conclusions

Our application-based setting has placed novel demands on relation extraction system trained with distant supervision, and in this paper we have shown that reasonable results can be obtained with only around 30 seed examples without requiring NER for pre-processing. Furthermore, we have demonstrated that learning entity filters and relation extraction jointly improves performance.

Acknowledgements

The research reported was conducted while the first author was at the University of Cambridge and funded by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 270019 (SPACEBOOK project www.spacebook-project.eu).

References

Razvan Bunescu and Raymond Mooney. 2007. Learning to extract relations from the web using minimal supervision. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 576–583.

Koby Crammer, Alex Kulesza, and Mark Dredze. 2009. Adaptive regularization of weight vectors. In *Advances in Neural Information Processing Systems 22*, pages 414–422.

Mark Craven and Johan Kumlien. 1999. Constructing biological knowledge-bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology*, pages 77–86.

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

He He, Hal Daumé III, and Jason Eisner. 2013. Dynamic feature selection for dependency parsing. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1455–1464, Seattle, October.

Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 2011. Knowledge-based weak supervision for information extraction of overlapping relations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 541–550.

Dan Klein and Chris Manning. 2002. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15*, pages 3–10.

Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*, pages 441–450.

Xiao Ling and Daniel S. Weld. 2012. Fine-grained entity recognition. In *Proceedings of the 26th Conference on Artificial Intelligence*, pages 94–100.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schtze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. 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 1003–1011.

Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *14th International Conference on Artificial Intelligence and Statistics*, pages 627–635.

Andreas Vlachos. 2012. An investigation of imitation learning algorithms for structured prediction. *Journal of Machine Learning Research Workshop and Conference Proceedings, Proceedings of the 10th European Workshop on Reinforcement Learning*, 24:143–154.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2010. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1013–1023.

Xingxing Zhang, Jianwen Zhang, Junyu Zeng, Jun Yan, Zheng Chen, and Zhifang Sui. 2013. Towards accurate distant supervision for relational facts extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 810–815, Sofia, Bulgaria, August. Association for Computational Linguistics.