spacebook-project.eu

Extraction of salient spatial descriptions from geo-tagged documents

Andreas Vlachos, Stephen Clark

Distribution: Public

SpaceBook
Spatial & Personal Adaptive Communication Environment: Behaviors & Objects & Operations & Knowledge
270019 Deliverable 3.2

January 28, 2014

Project funded by the European Community
under the Seventh Framework Programme for
Research and Technological Development

The deliverable identification sheet is to be found on the reverse of this page.

Project ref. no.	270019
Project acronym	SpaceBook
Project full title	Spatial & Personal Adaptive Communication Environment: Behaviors & Objects & Operations & Knowledge
Instrument	STREP
Thematic Priority	Cognitive Systems, Interaction, and Robotics
Start date / duration	01 March 2011 / 36 Months

Security	Public
Contractual date of delivery	M35 = January 2014
Actual date of delivery	January 28, 2014
Deliverable number	3.2
Deliverable title	Extraction of salient spatial descriptions from geo-tagged documents
Type	Report
Status & version	Final 1.0
Number of pages	13 (excluding front matter)
Contributing WP	3
WP/Task responsible	UCAM
Other contributors	
Author(s)	Andreas Vlachos, Stephen Clark
EC Project Officer	Franco Mastroddi
Keywords	Information Extraction, Distant Supervision

The partners in SpaceBook are:	**Umeå University**	UMU
	University of Edinburgh HCRC	UE
	Heriot-Watt University	HWU
	Kungliga Tekniska Högskola	KTH
	Liquid Media AB	LM
	University of Cambridge	UCAM
	Universitat Pompeu Fabra	UPF

For copies of reports, updates on project activities and other SpaceBook-related information, contact:

The SpaceBook Project Co-ordinator:
Dr. Michael Minock
Department of Computer Science
Umeå University
Sweden 90187
mjm@cs.umu.se
Phone +46 70 597 2585 - Fax +46 90 786 6126

Copies of reports and other material can also be accessed via the project's administration homepage, `http://www.spacebook-project.eu`

©2014, The Individual Authors

No part of this document may be reproduced or transmitted in any form, or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission from the copyright owner.

Contents

Executive summary

The goal of this part of Work Package 3 was to develop techniques for automatically populating the city model database by automatic analysis of text. In the Natural Language Processing community this task is known as Information Extraction (IE). An example would be taking a textual description of the Festival Theatre in Edinburgh and learning that it has a glass front; this information could then go into the database and be used by the SpaceBook system. Rather than focus on spatial descriptions, as the title of this report suggests, we first considered two relations which would be particularly useful for SpaceBook: the architects and completion dates of historical buildings. If this initial work was successful, we could then consider extending the IE system to extract additional relations, including relevant spatial descriptions.

The focus of this report is a description of the IE system based on a technique called *distant supervision*. Recent approaches following the distant supervision paradigm have focused on obtaining supervision for relation extraction by exploiting existing knowledge bases (KBs), from which they extract large sets of training seeds and entity lists, thus obviating the need for manual annotation. However, such KBs are not available for SpaceBook. Hence we have developed a novel IE approach that assumes only a small set of relation seeds and a search engine. We learn relation extractors following the techniques from [1] and evaluate them on two relations (architect name and completion year of buildings, as described above) and obtain good results using around 30 seeds. We also show how we improved the performance by learning entity filters jointly with the relation extractors using a technique from machine learning called *imitation learning*.

The report concludes with the difficulties we experienced in carrying out this research and some ideas for future work.

1 Introduction

The work in this report is related to the following task in Work Package 3:

> T3.2: Design and evaluate algorithms to extract relevant facts (e.g. visual descriptions) from geo-tagged texts over entities of interest for inclusion in the city data model

UCAM was the main partner responsible for carrying out this task, and we chose to see it as an example of *Relation Extraction*, and more broadly as an example of *Information Extraction (IE)*. Rather than focus on geo-tagged texts, as the task description suggests, we focused on readily available textual sources from the web (e.g. Wikipedia) and investigated whether information relevant to the SpaceBook city model could be automatically extracted from such sources.

Relation extraction between entities expressed in text is a popular natural language processing task applied to a variety of domains ranging from relations between persons and organizations [2] to biomedical interactions [3]. Relation extraction systems are commonly developed by training a supervised method on manually annotated data. While this paradigm has been successful, the need for annotated data limits its applicability in real-life applications.

In order to overcome this issue, recent work has focused on using existing knowledge bases (KBs) to generate annotated data that are used to train relation extraction systems, a paradigm commonly referred to as “distant” or “weak” supervision [4, 1]. Intuitively, relation tuples and entity lists are obtained from the KB and aligned with sentences from a text collection. If a sentence contains the entities forming a relation tuple, then the sentence becomes a positive relation extraction training instance for that relation, otherwise a negative one. While such automatically generated data contains noise, this paradigm has been shown to perform well [5, 6, 7].

A common characteristic of most relation extraction approaches in the distant supervision paradigm is that they consider as input a large KB, e.g. [1] reported using 7K-140K tuples per relation. Furthermore, they typically assume that the KB contains all the entities participating in the relations of interest, as they are used to construct the training and testing instances. However, it is often the case that such resources are unavailable in real-world applications. For example, in constructing a KB for SpaceBook we would like to extract information such as the architect of a historical building or the price range of food served in a particular restaurant, for which large sets of seeds and complete lists of entities are unlikely to be found in an existing KB.

In this work, we introduce a new framework for learning relation extraction, using only a small set of seeds and a search engine, which we believe places the task closer to the needs of real users, and makes it more appropriate and useful for the SpaceBook application. Using this framework, we learn extractors following the approach of [1] for two relations (architect name and completion year for historical buildings) and achieve good extraction performance using only 30 seeds.

Furthermore, we show how relation extraction performance can be improved by using entity filters as a first step in relation extraction. We avoid annotating training data for this task by learning an entity filter jointly with the relation extractor for each relation. For this purpose we use the imitation learning algorithm DAGGER [8], 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. Compared to the approach of [1], the jointly learned entity filters resulted in gains of 7 and 30 points in average precision for the completion year and the architect name relations respectively.

To summarise, we have found that the real-world setting of the SpaceBook application places new demands on a Relation Extraction system, which we have responded to by developing a new approach using the distant supervision paradigm.

In the remainder of this report we give an overview of our approach in Section 2. Section 3 describes how we apply distant supervision to the task, and how a pipeline of machine-learned classifiers can be used to decide if a particular pair is part of a particular relation (e.g. is Robert Adam the architect of Bute House?). Section 4 is a technical description of how we use imitation learning to learn the classifiers in the pipeline. Section 5 describes the experiments we ran to test the system, with empirical results. Section 6 describes how our new approach is related to existing work. Finally, Sections 7 and 8 provide some conclusions and suggestions for how the approach could be applied more broadly, beyond the two relations investigated, as a way of populating the SpaceBook city model database.

2 Approach overview

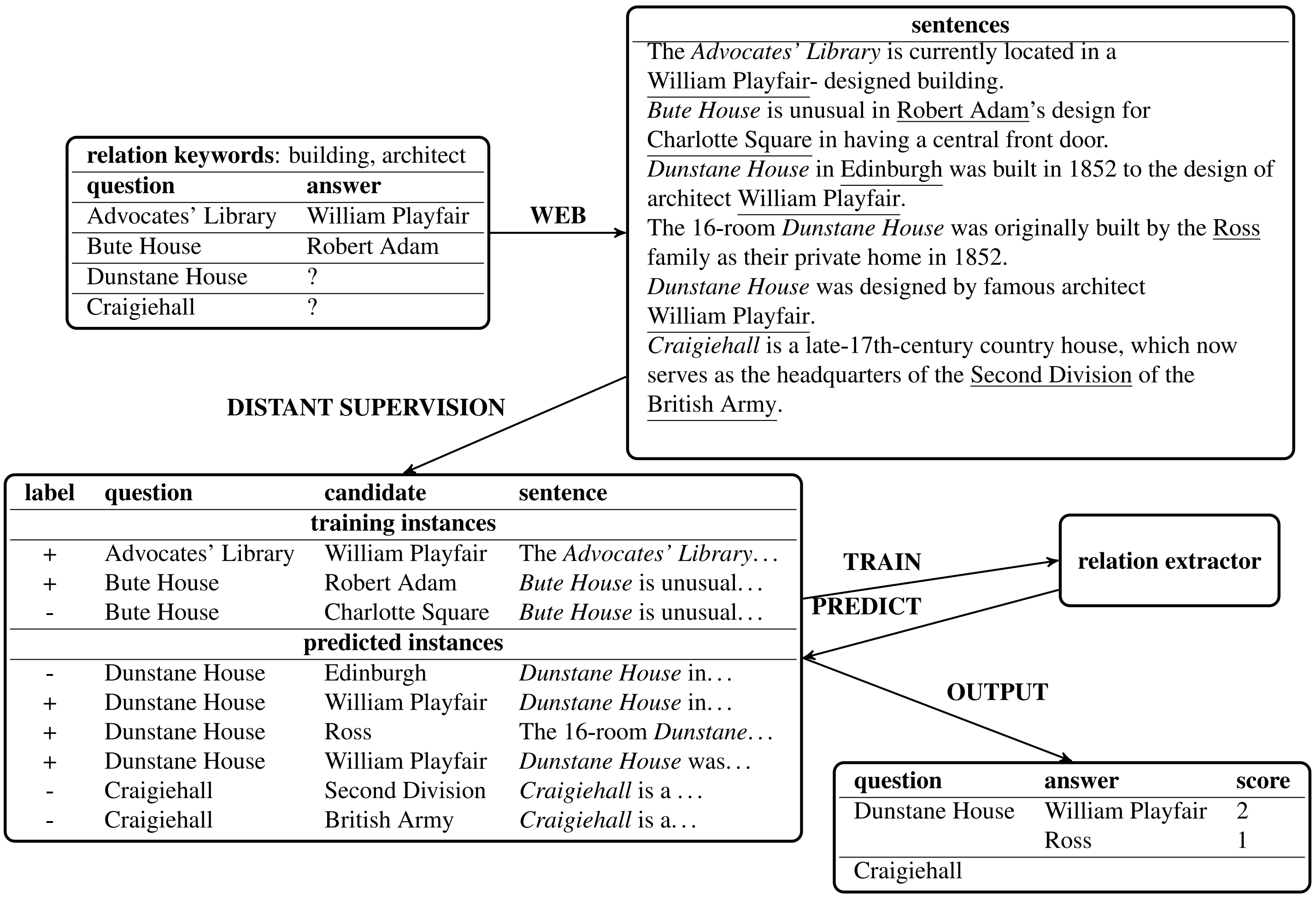
We will use the architect-building relation as an example to give an overview of our approach, as shown in Figure 1. The input to our approach consists of a few tuples and a set of keywords for each relation. Each tuple contains a question entity paired with the answer entity that is the correct answer for the relation in question. The keywords would be the search terms employed to find webpages likely to contain the answer using a search engine. The output of our approach is a system that, given an unseen question entity, returns the correct answer entity for the relation of interest. Figure 1 depicts the various stages of our approach for the architect name relation, in which the tuples "Advocates' Library - William Playfair" and "Bute House - Robert Adam" are the seeds given and the answers for "Dunstane House" and "Craigiehall" are sought.

We construct a training dataset for relation extraction in the following way. Initially, we query the web using a search engine combining each question entity with the relation keywords and build a text collection. In this process we ignore the answer entities, since they are not available when collecting texts for question entities during testing. We then extract the sentences mentioning the question entity. In each of these sentences, we recognize candidate answers using simple heuristics, for example if the answer entities to the relation are named entities then the candidate answers could be sequences of capitalized tokens.

Note that this is unlike recent work in distant supervision which assumes that both question and answer entities are available and commonly uses an existing named entity recognition system to align them with sentences in the text [7]. While we assume that the question entities are given, we rely on the keywords used in the search engine queries in order to extract sentences containing the question entities intended rather than synonymous ones. For example, the keyword "building" would help avoid extracting sentences containing other entities synonymous with a building of interest. Furthermore, since buildings are not among the standard named entity types recognized by existing systems, approaches relying on them would be unable to handle them well.

For each candidate answer, we generate an instance consisting of the sentence, the question entity and the candidate answer itself. We label each instance using the distant supervision assumption; i.e. if the candidate answer is the correct answer for the question entity then it is labeled as positive, otherwise as negative, thus creating annotated data to learn a relation extractor. Intuitively, the task defined can be stated as: "Does this sentence express that the candidate answer has the relation of interest with the question entity?" As expected, these labels are likely to be noisy; for example a sentence mentioning a building and its architect does not necessarily express the architect-of relation between them.

During testing, the same procedure to collect texts mentioning the question entity and extract instances is



relation keywords: building, architect

question	**answer**
Advocates' Library	William Playfair
Bute House	Robert Adam
Dunstane House	?
Craigiehall	?

sentences

The *Advocates' Library* is currently located in a William Playfair- designed building.
Bute House is unusual in Robert Adam's design for Charlotte Square in having a central front door.
Dunstane House in Edinburgh was built in 1852 to the design of architect William Playfair.
The 16-room *Dunstane House* was originally built by the Ross family as their private home in 1852.
Dunstane House was designed by famous architect William Playfair.
Craigiehall is a late-17th-century country house, which now serves as the headquarters of the Second Division of the British Army.

label	**question**	**candidate**	**sentence**
		training instances	
+	Advocates' Library	William Playfair	The *Advocates' Library*…
+	Bute House	Robert Adam	*Bute House* is unusual…
-	Bute House	Charlotte Square	*Bute House* is unusual…
		predicted instances	
-	Dunstane House	Edinburgh	*Dunstane House* in…
+	Dunstane House	William Playfair	*Dunstane House* in…
+	Dunstane House	Ross	The 16-room *Dunstane*…
+	Dunstane House	William Playfair	*Dunstane House* was…
-	Craigiehall	Second Division	*Craigiehall* is a …
-	Craigiehall	British Army	*Craigiehall* is a…

question	**answer**	**score**
Dunstane House	William Playfair	2
	Ross	1
Craigiehall		

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

followed for new question entities, with the exception that the instances are not labeled. For each question entity, the trained relation extractor is applied to all its instances and the candidate answers are ranked according to the number of instances containing them that were labeled positive by the classifier.

3 Learning relation extraction

Following [1], for each relation we learn a relation extractor using the positive and negative instances generated using the distant supervision assumption as described in the previous section. Each instance consists of a candidate answer, a question entity and a sentence, and we extract features describing the candidate answer and the relation between the candidate answer the question entity. The former include the tokens, their lemmas, information on their capitalization, the words and the bigrams preceding or following the candidate answer in the sentence, as well as the lemmas of the words one syntactic dependency away from it conjoined with the dependency. The latter include the lexicalized dependency path from the question entity to the candidate answer, as well as the lemmas on this path.

To learn the binary classifier for each relation we implemented the adaptive regularization of weight vectors (AROW) algorithm [9]. AROW is an online algorithm for linear predictors, thus it scales easily to large datasets. In addition, it takes into account the rarity of each feature and adjusts the per-feature learning rates so that popular features do not overshadow rare but useful ones. Furthermore, it can handle non-separable data, which is likely to be useful given the noise due to the distant supervision assumption. Since we are operating in a batch learning setting (i.e. we have access to all the training examples and their order is not meaningful), we perform multiple rounds over the training examples randomly shuffling their order, and average the weight vectors. Finally, to further reduce the effects of noise, we removed features appearing only once in the training data.

As we do not assume a list of entities, the task becomes harder than the one commonly tackled in distant supervision approaches, since the candidate answers are very often inappropriate for the relation at question. Therefore, it would be beneficial to have an entity filter tailored to the relation of interest, e.g. a filter that would accept “William Playfair” (a famous architect) but reject “Ross” (the owners) in Figure 1. Since labeling data for every new answer entity type to learn such filters would be impractical, we learn it jointly with relation extraction using imitation learning, which will be discussed in detail in the following sections.

4 Imitation learning

Imitation learning algorithms aim at learning controllers from demonstrations by human experts [10, 11]. Unlike standard reinforcement learning algorithms [12], they do not require the specification of a reward function by the user. Instead, the algorithm observes a human expert performing a sequence of actions to predict instances of the task in question and learns a policy that “imitates” the expert with the purpose of generalizing to unseen data. These actions have dependencies between them, since earlier ones affect the ones following them and successful imitation algorithms learn how to take them into account.

Imitation learning algorithms such as SEARN [13] and DAGGER [8] have been applied successfully to a variety of tasks such as summarization, biomedical event extraction and dynamic feature selection [13, 14]. In this work we focus on DAGGER as it has been shown to be more stable than SEARN [8] and highlight its ability to handle missing labels in the training data.

4.1 DAgger

The dataset aggregation (DAGGER) algorithm [8] forms the prediction of an instance s as a sequence of T actions $\hat{y}_{1:T}$ predicted by a learned policy which consists of one or more classifiers. Each action $\hat{y}_t$ can use features from s and all previous actions $\hat{y}_{1:t-1}$, thus exploiting possible dependencies. The number of actions T taken in predicting each instance is not defined in advance but it depends on the actions chosen.

During training, DAGGER converts the problem of learning how to predict sequences of actions into cost sensitive classification (CSC) learning. In CSC each training example has a vector of misclassification costs associated with it, thus rendering some mistakes on some examples to be more expensive than others [15]. The dependencies between the actions are learnt by appropriate generation of CSC examples. In particular, the features for each action y_t can take into account all previous actions $\hat{y}_{1:t-1}$, while the costs for each possible action c_t take into account the effect of y_t on the remaining sequence of actions. The

Algorithm 1: Imitation learning with DAGGER

```
Input: training instances $\mathcal{S}$, expert policy $\pi^\star$, loss function $\ell$, learning rate $\beta$, CSC learner $CSCL$
Output: Learned policy $H_N$
1  CSC Examples $E = \emptyset$;
2  for $i = 1$ to $N$ do
3      $p = (1-\beta)^{i-1}$ ;
4      current policy $\pi = p\pi^\star + (1-p)H_{i-1}$ ;
5      for $s$ in $S$ do
6          Predict $\pi(s) = \hat{y}_{1:T}$;
7          for $\hat{y}_t$ in $\pi(s)$ do
8              Extract features $\Phi_t = f(s, \hat{y}_{1:t-1})$ ;
9              foreach possible action $y_t^j$ do
10                 Predict $y\prime_{t+1:T} = \pi(s; \hat{y}_{1:t-1}, y_t^j)$ ;
11                 Estimate $c_t^j = \ell(\hat{y}_{1:t-1}, y_t^j, y\prime_{t+1:T})$ ;
12             Add $(\Phi_t, c_t)$ to $E$;
13     Learn $H_i = CSCL(E)$ ;
```

cost for each possible action y_t^j is estimated by assuming that the action y_t^j was taken; then the following actions for that instance $y_{t+1:T}$ are predicted using the learned policy H_i, and the whole sequence of actions is compared against the correct output for that instance.

Algorithm 1 presents the training procedure of DAGGER in more detail.DAGGER requires a set of labeled training instances $\mathcal{S}$ and a loss function ℓ that compares complete action sequences for instances in $\mathcal{S}$ against the correct output for them. In addition, an *expert policy* $\pi^\star$ must be specified which is a function that returns the optimal action $\hat{y}_t$ for the instances in the training data, which is akin to an expert demonstrating the task. $\pi^\star$ is typically derived from the labels of the training instances; for example in part of speech tagging $\pi^\star$ would return the correct tag for each token. In addition, the learning rate β and a CSC learner (*CSCL*) must be provided. The algorithm outputs a learned policy H_N that, unlike the expert policy $\pi^\star$, can generalize to unseen data.

Each training iteration begins by setting the probability p (line 3) of using $\pi^\star$ in the current policy π. In the first iteration only $\pi^\star$ is used but in later iterations π becomes stochastic as for each action we use $\pi^\star$ with probability p and the learned policy from the previous iteration h_{i-1} with probability $1-p$ (line 4). Then π is used to predict each training instance s (line 6). For each action $\hat{y}_t$, a CSC example is generated (lines 7-12). The features Φ_t are extracted from s and the previous actions $\hat{y}_{1:t-1}$ (line 8) and are expected to be useful in predicting the current action $\hat{y}_t$. For example, in part-of-speech tagging, commonly used features include the word we are predicting, the words preceding it and following it, as well as the tag predicted for the previous word. The cost for each possible action y_t^j is estimated by predicting the remaining actions $y\prime_{t+1:T}$ needed for s using π (line 10) and calculating the loss incurred given y_t^j w.r.t. the correct output for s using ℓ (line 11). The features together with the costs for each possible action form a CSC example (Φ_t, c_t) (line 12). At the end of each iteration the CSC examples obtained from all iterations are used by the CSC learning algorithm to learn the classifiers for a new H_i (line 13).

When predicting the training instances (line 6), the policy learned in the previous iteration H_{i-1} is used, thus the CSC examples used to learn H_i in the current iteration depend on the predictions of H_{i-1}. The degree to which it is used depends on the probability p set in the beginning of each iteration. By gradually decreasing the use of the expert policy in the current policy, the policy is adjusted to its own predictions, thus learning how to predict sequences of actions jointly.

The learning rate β determines how fast the current policy π moves away from $\pi^{\star}$. A special case is obtained when $\beta = 1$, also referred to as pure policy iteration or parameter-free version of the algorithm. In this case, $\pi^{\star}$ is used only in the first iteration and in the following iterations π uses only the policy from the previous iteration H_{i-1}. This setting renders DAGGER training deterministic (π is no longer stochastic), but it also becomes harder due to the abrupt transition from the expert to the learned policy. [8] showed that DAGGER can perform well even in this case and we consider this parameter-free version for the remainder of this paper.

4.2 Handling missing labels

As mentioned in the previous section, the loss function ℓ in DAGGER is only used to compare complete action sequences against the correct output. In other words, it does not need to decompose over the actions in the sequence in order to evaluate them independently. Since we only need to evaluate complete action sequences, this gives us the option to use loss functions that ignore actions which are taken as intermediate steps in predicting an instance. For example, in a sequential tagging task (e.g. part-of-speech) we could use a loss function that only takes into account the tag predicted for the last token in a sentence, even though the prediction of this tag is dependent on the predictions for the previous tokens.

The use of non-decomposable loss functions in DAGGER can be very useful when the training data has missing labels. When generating a CSC training example (lines 10-11), we do not need to know whether an action y_t^j is correct or not, we just need to evaluate what the effect of y_t^j is on the loss incurred by the complete action sequence. Thus, the classifiers forming H_i are trained to predict actions that minimize the loss on the instance, independently of what the correct action might have been.

The only component in DAgger for which knowledge of all the correct actions is assumed is the expert policy $\pi^{\star}$ for the training instances. As explained above, $\pi^{\star}$ is used progressively more sparingly as the learning progresses, and in the parameter free version of DAgger we consider in this work it is only used in the first iteration. Given its limited impact on the learning, we define $\pi^{\star}$ to be returning randomly or heuristically chosen actions that allow it to predict the instance correctly w.r.t. the loss function. Returning to the sequential tagging example, assuming that the tags for all the tokens are missing except for the last token in the sentence, the expert policy could return a random action for all tokens apart from the last one, for which the (known) correct tag must be returned.

4.3 Joint entity filtering and relation extraction

In this section we describe how we learn the entity filter described in Section 3 jointly with relation extraction using DAGGER. Each instance consists of a candidate answer, a question entity and a sentence, and we decompose its prediction in two stages: first we apply the entity filter and decide whether the candidate answer is of the correct entity type for the relation in question, and then we apply the relation extractor to decide whether the sentence expresses this relation between the answer and the query entity.

If the prediction for the first stage is negative (i.e. the candidate answer is not of the correct type), then the second stage is not reached as the prediction for the instance is negative by definition (i.e. the relation is not expressed in the sentence).

Following the description of DAgger in Section 4, the policy H_i learned in each iteration consists of a separate binary classifier for each stage. As discussed in Section 3, we have labels for relation extraction obtained by distant supervision, but not for the entity filter. Therefore, we take advantage of the capacity of DAgger to learn with non-decomposable loss functions and define ℓ to be 0 in case the prediction for the relation extraction stage is correct and 1 otherwise. We define the expert policy $\pi^\star$ for the missing entity filter labels heuristically, by assuming that all candidate answers are of the correct type. While far from perfect, defining the expert policy in this way forces DAgger to search for instances in which this assumption leads to incorrect relation extraction predictions and learn from them a suitable entity filter. For the relation extraction stage, the expert policy returns the labels obtained by distant supervision.

Since the losses returned are either 0 or 1, the CSC learning task is equivalent to ordinary binary classification learning. Therefore we use the AROW classification learning algorithm, as in our re-implementation of the approach of [1]. For the entity filter we extract the features describing the candidate answer and for the relation extractor the features describing the relation between the candidate answer and the question entity. Using the same features and classifier ensures a fair comparison between the two approaches.

5 Evaluation

For evaluation we used two relations, architect name and completion year for historical buildings in the city of Edinburgh. For each of the 138 listed historical buildings mentioned in Wikipedia[1] we used a search engine to look for the answer entities for the relations in question. In case an answer could not be found or if conflicting answers were found in the first page of results we omitted the building from the answer key of that relation. The latter occurred relatively often, since historical buildings such as churches and castles are likely to be built and rebuilt at different points in time by different architects. The assumption that there exists only one correct answer per instance is only employed to facilitate the evaluation; our approach can return multiple answers as discussed in Section 2. As a result, we obtained 60 completion year tuples and 68 architect name tuples. It is worth pointing out that while the building names were obtained from Wikipedia, many of the answers were not on Wikipedia but on other websites, thus highlighting the difficulty of the dataset.

Recall that the system can provide more than one answer for each question entity; hence the answers are 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 the 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 returned over the total number of question entities, and precision as the chance of finding the correct answer while traversing those returned. Finally, in the *all* mode we evaluated precision at all recall points by varying the thresholds used in the respective classifiers and, following [7], we report average precision (AP) [16]. This evaluation measure provides an assessment of how well a system trades precision for recall.

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

5.1 Data preprocessing

The first stage in our approach described in Section 2 is to collect documents from the web using a search engine. The queries are formed by combining each question entity (historical building in our case) with the keywords provided for the relation of interest. These are submitted to Bing via its Search API service and the top 300 results for each query are obtained. We download the pages linked in the results and extract their textual content with BoilerPipe[2][17].

We then process the texts to extract sentences containing the question entities and candidate answers using components of the Stanford CoreNLP toolkit[3]. We first tokenize the text contained in each page and split it into sentences. Then we try to match the question entity with tokens in each of the sentences, allowing for minor differences in tokenization, whitespace and capitalization. If a sentence contains the question entity, we parse it using the parser of [18] and look for candidate answers of two types, depending on the relation: single-token numbers (candidate answers for relations involving years, monetary sums, etc.) and consecutive tokens tagged as proper names by the part-of-speech tagger (candidate answer for relations involving named entities, e.g. persons, countries, streets, etc.). Other types of candidate answers can be supported, as long as they span contiguous tokens and can be extracted using string matching or heuristics based on syntactic information.

5.2 Results

We first perform web data preprocessing (Sec. 5.1) obtaining 974K and 4.5M labeled instances for the completion year and the architect name relation respectively. The datasets are highly imbalanced (i.e. most sentences do not express the relation in question) and in both relations there are question entities for which no positive instance was generated. We did not remove these entities from the evaluation since we consider the web data preprocessing to be part of our system and we want to be able to measure the impact of future improvements in this component. Furthermore, for many buildings the correct answer was mentioned in very few sentences (sometimes none), and Wikipedia did not contain any such sentences, facts indicative of the difficulty of the task. Note that such rare entities are commonly ignored in recent distant supervision approaches.

We split the data for both relations in development and testing, each containing half of the instances. All development and feature engineering was done using 4-fold cross-validation on the former, while testing was used only to report results. For the two-stage relation extraction system (henceforth 2stage) we used 12 training iterations with DAgger. Since AROW training is stochastic due to the random shuffling of its training examples, we average the results over three runs.

We ran experiments with three systems; our re-implementation of [1] described in Section 3 (henceforth Mintz), the relation extraction approach jointly learned with the entity filtering using imitation learning described in Section 4 (henceforth Imit) 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).

Results by all models on the test data 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

[2]`http://code.google.com/p/boilerpipe/`

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

	R *top*	P *top*	F *top*	R *all*	P *all*	F *all*	AP *all*
Base	0	0	0	62	0.2	0.4	-
Mintz	15	26	19	23	17	20	21
Imit	26	65	37	30	55	39	51
Base	28	28	28	90	10	18	-
Mintz	52	71	60	67	68	67.5	69
Imit	50	68	58	67	67	67	76

Table 1: Results for the 3 systems on building-architect (top) building-completion_year (bottom).

in completion year extraction, but it fails completely in architect name. This is due to the existence of many other names 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 given the data preprocessing, 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 very low precision.

Both the machine-learned models improve upon Base substantially on both datasets, with the Imit model achieving the best performance with a wide margin in architect name extraction. The gains are particularly pronounced in terms of precision, with average precision being 30 points higher. In completion year extraction the differences are smaller, with Mintz being slightly better than Imit. These small differences are expected since recognizing completion years is much easier than recognizing architect names, thus learning an entity filter for them is less likely to be useful. Nevertheless, it helps Imit to achieve a better precision-recall trade-off, as its average precision is seven points higher. Furthermore, inspection of the weights learned for the entity filter showed that it had learned some useful distinctions, for example that 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 disagreements at the sentence level between the predictions of Imit and the noisy labels obtained by distant supervision. We found that Imit frequently predicted the relation correctly, but the answer was incorrect due to multiple entities with the same name. This is in addition to the noisy positive labels commonly discussed in the distant supervision literature. Also, in many cases Imit failed to predict correctly positive instances due to feature sparsity, suggesting it as a direction for future work.

6 Related work

The most closely related relation extraction learning approach to ours was proposed by [19]. While they also learned a relation extractor using a few tuples and a search engine as input, the sentences were collected from the web via queries containing both the question and the answer entities from the input tuples, while we use the question entities only for this purpose. Such an approach would not be applicable in the case where a list of candidate answers is unavailable, or inefficient if the list is large, as is the case with relations studied in our experiments. Furthermore, it is unclear whether an extractor trained on sentences restricted to contain certain candidate answers would generalize well to others. On the other hand, our proposed architecture collects and labels sentences without making this assumption and learns an appropriate relation extractor.

Another approach that learns from a small set of tuples is Snowball [20]. However, it requires NER components to identify the arguments of the relation of interest. Furthermore, the authors found that most of the errors in their experiments were due to NER errors, thus stressing the importance of this requirement. A similar approach by [21] does not require NER, but their evaluation is restricted to the year of birth relation, for which answer recognition is not crucial to success, as we also found in our experiments with the completion year relation.

Other related approaches include systems developed under the open information extraction paradigm [22]. While many of these systems do not assume any manually annotated data (e.g. TextRunner [23] obtains its supervision from the output of a syntactic parser), the relations extracted are labeled not according to a specification provided by the user but using words from the sentence in which they were found. Thus, if the goal is to find answers for a particular relation which is the case in this work, the user would still need to define a mapping from the labels of the extracted relations to the relation of interest. The never-ending language learner [24] uses a seed ontology to provide labeled relations but the order in which answers to particular relations and entities are extracted is not dictated by the user but by the learning algorithm. In contrast, our approach allows the user to specify the entities and the relations to which answers are needed. A disadvantage though is that it cannot identify new entities to extract answers for.

7 Conclusions

In this work we introduced a new framework for learning relation extraction using only a small set of seeds and a search engine. Unlike most previous work it does not rely on a large existing KB to extract lists of entities participating in the relations. Furthermore, we show how to use imitation learning in order to learn an entity filter jointly with relation extraction, even though we did not have labels for the former. We evaluated our approach on two relations and obtained good results using around 30 tuples. Furthermore, we demonstrated that learning the entity filter jointly with relation extraction outperforms a typical distantly supervised approach using the same features and classifier. We believe that given its minimal requirements the proposed framework is better suited to the needs of real users. In addition, the application of imitation learning to tasks with missing labels is likely to be of interest to other tasks facing this issue.

8 Difficulties and Future Work

The main cost in running the relation extraction framework is the manual creation of the seed examples. For example, for the work described above, the first author had to manually create two lists containing pairs of historical buildings and their architects and historical buildings and their years of completion. For both relations this involved time-consuming work reading pages from the internet (despite both lists being relatively short). It was this bottleneck in creating the seed lists which prevented us from applying the system to additional relations, such as spatial descriptions.

In order to address this issue the two authors supervised an undergraduate Part II project in the Computer Laboratory looking at the use of crowd-sourcing to create the initial seed lists. The project was carried out by Joaquim dSouza, and employed the crowd-sourcing service Crowdflower. The results were highly promising: for relations including films-directors, directors-birth place, states-ruler, and rulers-date of abdication, the system was able to elicit high-quality lists with small amounts of time and monetary costs.

If the system were to be fully deployed within the SpaceBook application, the obvious next step would be to use the crowd-sourcing system to elicit many lists of seed relations for relations relevant to SpaceBook, and then run the relation extraction system using these seeds as input. In principle this would allow many of the relations in the SpaceBook city model database to be populated automatically with very little manual intervention (beyond the manual annotation carried out by the crowd-source workers).

References

[1] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 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, 2009.

[2] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The Automatic Content Extraction (ACE) Program – Tasks, Data, and Evaluation. In *Proceedings of the Fourth Conference on Language Resources and Evaluation*, 2004.

[3] Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, and Jun'ichi Tsujii. Overview of BioNLP Shared Task 2011. In *Proceedings of the BioNLP 2011 Workshop Companion Volume for Shared Task*, pages 1–6, 2011.

[4] Mark Craven and Johan Kumlien. 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, 1999.

[5] Raphael Hoffmann, Congle Zhang, Xiao Ling, Luke Zettlemoyer, and Daniel S. Weld. 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, 2011.

[6] Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D. Manning. Multi-instance multi-label learning for relation extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 455–465, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[7] Sebastian Riedel, Limin Yao, Benjamin M. Marlin, and Andrew McCallum. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, GA, 2013.

[8] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. 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, 2011.

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

[10] Pieter Abbeel. *Apprenticeship Learning and Reinforcement Learning with Application to Robotic Control*. PhD thesis, Department of Computer Science, Stanford University, 2008.

[11] Umar Syed. *Reinforcement learning without rewards*. PhD thesis, Department of Computer Science, Princeton University, 2010.

[12] Richard Sutton and Andrew Barto. *Reinforcement learning: An introduction*. MIT press, 1996.

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

[14] Andreas Vlachos. 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, 2012.

[15] Pedro Domingos. Metacost: a general method for making classifiers cost-sensitive. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining*, pages 155–164. Association for Computing Machinery, 1999.

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

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

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

[19] Razvan Bunescu and Raymond Mooney. 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, 2007.

[20] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, pages 85–94, San Antonio, TX, USA, 2000. Association for Computing Machinery.

[21] Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Names and similarities on the web: fact extraction in the fast lane. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 809–816, 2006.

[22] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Communications of the ACM*, 51(12):68–74, December 2008.

[23] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2670–2676, 2007.

[24] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence*, 2010.