Universal Schema for Slot Filling and Cold Start: UMass IESL at TACKBP 2013

Sameer Singh, Limin Yao, David Belanger, Ari Kobren, Sam Anzaroot, Michael Wick, Alexandre Passos, Harshal Pandya, Jinho Choi, Brian Martin, Andrew McCallum
School of Computer Science
University of Massachusetts, Amherst
mccallum@cs.umass.edu

Abstract

We employ *universal schema* for the TAC KBP slot filling and cold start tasks. The technique enlarges the set of relations in an ontology, e.g., TACKBP slots, to contain all surface patterns between pairs of entities in a large corpus. By factorizing the matrix of co-occurrences between entity pairs and universal schema relations, we are able to predict new target slots. This differs fundamentally from traditional relation extraction approaches because an entire knowledge base is constructed jointly over train and test data. To produce submissions for the slot filling and cold start tasks, we simply query this knowledge base. We describe universal schema, our data preprocessing pipeline, and additional techniques we employ for predicting entities' attributes.

1 Introduction

Due to its importance in information extraction pipelines, relation extraction has attracted attention in TAC KBP for a number of years in the Slot Filling track, and as a vital sub-task of the recently-introduced Cold-Start track.

Many existing relation extraction techniques (Liu and Zhao, 2012; Min et al., 2012; Roth et al., 2012) perform the following: (1) matching of textual mentions to the query mentions, often utilizing information retrieval techniques, (2) extraction of the context around the mention, (3) prediction of the relation being expressed in the context (if any), and (4) aggregation of the individual classifications to resolve redundancies and inconsistencies. Although this overall architecture is common, techniques used for matching, extraction, relation identification (using sophisticated rules, lexicons, classifiers, graphical models, etc.) and aggregation vary considerably.

There are a number of disadvantages, unfortunately, inherent in such traditional systems. The foremost problem is obtaining training data for learning the extractors, as the amount of training data available as part of KBP is limited. Many systems use manually-created seed patterns to bootstrap a labeling (often in combination with the TACKBP training data), followed by induction to expand the set of training examples for the extractors. More recently, *distant supervision* approaches, which use the relations in external knowledge bases such as Freebase or Wikipedia (manually aligned to the TACKBP schema), have achieved noteworthy accuracy (Roth et al., 2012). However, the resulting training data from using either seed patterns or distant supervision is often low-quality due to noise in the process of producing training data.

Further, it is often not the case that a single pattern in a sentence is indicative of a relation between two entities, and instead, multiple relations/patterns may in combination imply a relation that is not evident when the patterns are observed in isolation. For example, recognizing that *Apple Inc.* is headquartered in *California* might require an extraction that provides evidence that *Apple* is located in *Palo Alto*, and a separate extraction that indicates *Palo Alto* is in *California*. Along with requiring inference of relations across multiple extractions, this example further requires extraction of relations for non-query entities (*Palo Alto* in this example) and also would benefit from leveraging the implicit implications between relations.

In response, we introduce a novel approach to the TAC KBP slot filling and cold start tasks that produces answers jointly about all entities and relations, rather than on a query-by-query or per-extraction basis. From a large collection of documents consisting of both training and test KBP documents, we construct a knowledge base over all the entities (including many that do not appear in the queries or the reference knowledge base). Relations extraction is performed using the matrix factorization-based universal schema approach (Riedel et al., 2013) that learns correspondences between co-occurring entity pairs, observed textual patterns, and the labeled relations, as part of a joint optimization over training and test data. Due to the nature of the factorization, the model is able to produce confidence values for its outputs, and generalize to entities for which we observe surface patterns but were not linked to reference knowledge base entities.

The overall system consists of the following architecture. First, we process the text corpus using a natural language processing pipeline. This includes part-of-speech tagging, dependency parsing, mention finding and coreference, as described in Section 2. Second, we extract binary relations between pairs of entities using universal schema (Section 3). In order to predict the string-valued slots (such as `alternate_name`) that are not supported by *universal schema*, we use manually constructed rule-based heuristics (Section 4). The extractions from two steps are combined and resolved as described in Section 5 to produce an internal knowledge base about the corpus.

Since this was our first year participating in the TAC KBP Slot Filling and Cold Start tasks, we do not expect to outperform the existing approaches. In Section 6, we investigate our predictions on the KBP 2013 tasks, and demonstrate the various associations between surface patterns and schema relations that our model learns. According to the official evaluation that scores the correctness of extracted provenances, we obtain 12.5% F1 for the entity-valued slots and 19.3% for the string-valued slots. Further, we observe that our universal schema extraction method achieves high recall, but suffers from low precision. Conversely, our rule-based extractors produce high precision and low recall. Overall, we achieve 13.7% F1 for slot filling, and 7% F1 for cold start.

2 Data Processing Pipeline

Figure 1 outlines our processing pipeline. The pipeline is nearly identical for both our slot filling and cold start systems; it differs only in the set of documents and queries that are used, and the filtering, if any, of the resulting knowledge base in the final stage. For slot filling, we are given a list of queries, which we issue to our knowledge base. For cold start, we query for every relation for all the entities that appear in cold start documents.

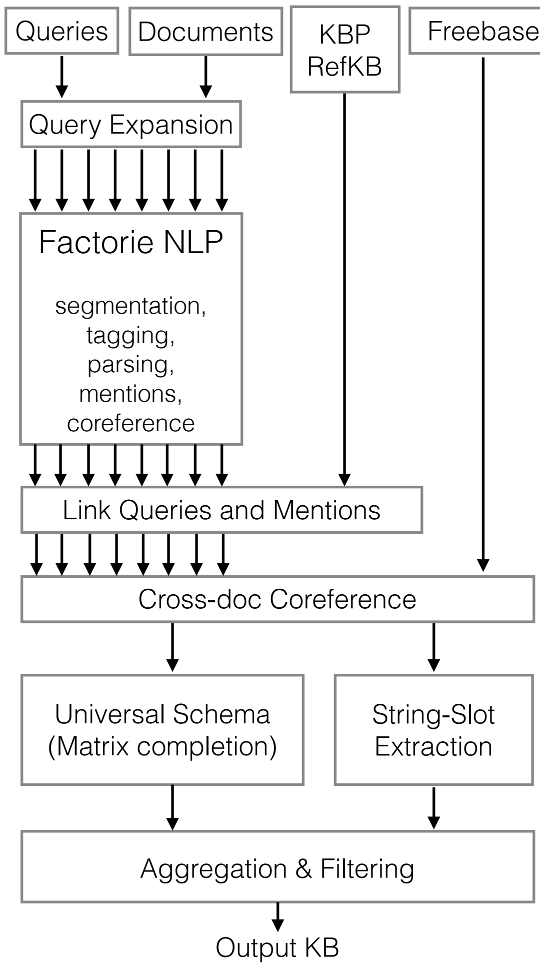


Figure 1: **UMass IESL Pipeline:** Earlier stages process each document independently. However, in later stages, the pipeline treats relation extraction as joint inference of the output KB.

2.1 Corpus Selection

Recall that universal schema performs relation extraction jointly on train and test data. Therefore, we need to assemble a sub-collection of the TAC KBP training documents that are relevant either for training or for answering the queries. For slot filling, we select $120K$ documents by performing query expansion on the 2013 queries (both training and test) using the Galago index on all the KBP source documents (Cartright et al., 2012), used for entity linking in TAC KBP 2012 (Dietz and Dalton, 2012). For the cold start track, we append the the cold start source documents to this collection.

2.2 Natural Language Processing

We employ the open-source FACTORIE package (McCallum et al., 2009) as our natural language processing pipeline for processing the documents. This software package includes a number of state-of-the-art machine learning models for NLP processing, including part-of-speech, dependency parsing, named entity recognition (NER), and within-document coreference.

2.3 Mention Finding and Entity Discovery

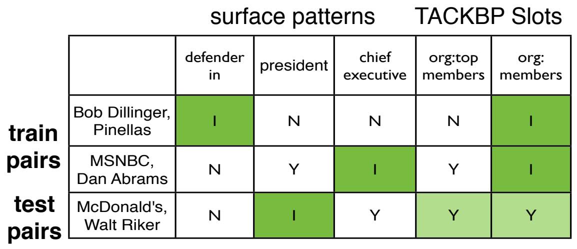
Mention finding and entity discovery play a critical role in our pipeline, since without accurate entities, we cannot build a high quality knowledge base to answer the TACKBP slot filling and cold start queries. Our set of mentions consist of the detected named entities and pronouns, as well as all queries and annotations. We use the FACTORIE within-document coreference system to cluster mentions for each document into anaphoric sets. This system performs greedy left-to-right grouping of mentions with features based on Bengtson and Roth (2008).

To assemble the cross-document entities for the knowledge base, all the within-document entities, along with entities from the reference KB, need to be aggregated. The observed *string names* of the within-document and the reference entities are normalized by removing prefixes and suffixes such as titles and honorifics, and expanding acronyms, followed by a string-matching based grouping. For cold-start, we also include relation annotations from Freebase (described in the next section) for which we perform trivial linking of Freebase entities and our cross-document entities based on available names and redirects in Freebase.

3 Slot Filling Using Universal Schema

We cast slot filling as relation extraction, considering each slot as a relation instance of the query and the slot value. We employ universal schema for slot filling (Riedel et al., 2013). Instead of classifying entity pairs into pre-defined relation types, universal schema takes both the surface patterns and pre-defined types as relations, and uses matrix factorization to discover implications among them. This approach can leverage unlabeled data, while avoiding brittle alignment errors in the distant supervision methods that have been popular in recent TAC KBP submissions.

We fill a matrix with relation instances, where



		surface patterns			TACKBP Slots	
		defender in	president	chief executive	org:top members	org: members
train pairs	Bob Dillinger, Pinellas	I	N	N	N	I
	MSNBC, Dan Abrams	N	Y	I	Y	I
test pairs	McDonald's, Walt Riker	N	I	Y	Y	Y

Figure 2: **Illustration of universal schema.** Each row represents an entity pair and each column a relation. Observed cells (green cells marked "1") denote that an entity pair co-occurs with a relation in text or in TACKBP annotation. The rest of the cells are unobserved during training, and are filled with oracle decisions on whether the fact holds. Our goal is to predict whether a relation (slot) is true for particular entity pairs. The last row is an example test pair, we predict two slots for this pair (light green cells).

each row corresponds to an entity pair and each column to a relation, including surface features and slot names. The surface features include the sequence of words between two entity mentions, the dependency path connecting two mentions, NER tags of the mentions, and so on. We only include TACKBP slots that describe relations between two entities (i.e. the *entity-valued* slots). Our goal is to predict target slots for the entity pairs. Matrix factorization, that provides a low-dimensional embedding for each row (entity pair) and column (slot or surface pattern), is used to discover implicature among surface features and slots.

Before we describe our model and algorithm, we illustrate how universal schema works in Figure 2. During training, we learn low dimensional embeddings for entity pairs and relations using observed patterns, which can be used to complete unobserved cells of the matrix. As shown in the example, we would like to learn that `org:top_members` implies `org:members`, "president" is indicative of `org:top_member` only if `org:members` holds, and so on. Once we learn the embeddings of the rows and columns, we query the scores of the relevant cells, and picks the confident ones to be included into the output KB. For slot filling we only query cells that have a KBP query mention as part of the entity pair and one of the KBP slots as the column.

3.1 Matrix Factorization

For matrix factorization, we embed each entity pair (row) e and relation/feature (column) r as latent vectors $\mathbf{a}_e$ and $\mathbf{v}_r$ in a K-dimensional space, respectively. To model each binary entry in the matrix, we use a logistic regression version of matrix factorization that is appropriate for binary matrices (Collins et al., 2001). Thus model for each cell $x_{e,r}$ is as follows:

$$\begin{aligned}\theta_{e,r} &= \sum_c a_{e,c} v_{r,c} \\ x_{e,r} &= \sigma(\theta_{e,r}) \\ \text{where, } \sigma(\theta) &= \frac{1}{1+\exp(-\theta)}\end{aligned}$$

The first formula factorizes the matrix into multiplication of two smaller matrices. The second formula applies the logistic function to the θ scores obtained from factorization to model a binary cell. This has a probabilistic interpretation: each cell is drawn from a Bernoulli distribution with natural parameter θ. Note that an observed pattern or slot r between an entity pair e is encoded by $x_{e,r} = \sigma(\theta_{e,r}) \equiv 1$.

In the following, we describe how we train our model. We learn low dimensional representations for entity pairs and relations by maximizing the log likelihood of the observed cells under the probabilistic model above. Note that in our training data we only observe positive cells and have no accurate data on which relations do *not* hold for an entity. However, learning requires negative training data. We address this issue by sampling unobserved relations for an entity based on their frequencies in the whole dataset and treating them as negative. The joint probability of all cells is defined as:

$$\begin{aligned}&\prod_{e,r} p(x_{e,r}=1)^{\delta(x=1)}(1-p(x_{e,r}=0))^{\delta(x=0)} \\ &= \prod_{e,r} \sigma(\theta_{e,r})^{\delta(x=1)}(1-\sigma(\theta_{e,r}))^{\delta(x=0)}\end{aligned}$$

For simplicity, we elide the subscript of each cell, and $\delta(x=1)$ is the number of observed cells.

To further simplify the joint probability, we represent negative cells as positive by choosing a different natural parameter. Thus the joint probability becomes $\prod_{e,r} \sigma(\theta_{e,r})$. Incorporating regularization and expanding terms, log-likelihood training can be described as the following optimization:

$$\underset{\theta}{\text{argmax}} \sum_{(e,r)} \log \sigma\Big(\sum_c a_{e,c} v_{r,c}\Big) - \lambda(||\mathbf{a}||_2^2 + ||\mathbf{v}||_2^2)$$

We use stochastic gradient optimization to effectively deal with the large scale of our matrices. In each iteration, we traverse random permutations of all training cells, randomly sample 3 to 5 negative cells for each training cell, and update the corresponding $\mathbf{a}_e$ and $\mathbf{v}_r$ vectors for the positive and negative cells based on their corresponding gradients. We update the parameters of a positive cell (e, r) using the following formulas, iterating over each component c, with learning rate l:

$$\begin{aligned}a_{e,c} &\mathrel{+}= l(1-\sigma(\theta))v_{r,c} - \lambda a_{e,c} \\ v_{r,c} &\mathrel{+}= l(1-\sigma(\theta))a_{e,c} - \lambda v_{r,c}\end{aligned}$$

Likewise for a negative cell, we update the parameters using:

$$\begin{aligned}a_{e,c} &\mathrel{+}= l(0-\sigma(\theta))v_{r,c} - \lambda a_{e,c} \\ v_{r,c} &\mathrel{+}= l(0-\sigma(\theta))a_{e,c} - \lambda v_{r,c}\end{aligned}$$

For confidence of our prediction, we use its score, i.e. $x_{e,r} = \sigma(\theta_{e,r})$.

3.2 Observed Cells

The above approach learns dependence among the columns of the matrix, which, in our case, consists of observed surface patterns and relations described in this section.

We use the dependency path of the sentence and the word sequence between two entity mentions as surface patterns. Additionally, we use conjunctions of trigger words and entity types (see Table 1 for examples). A dependency path is a concatenation of dependency relations and words on the path connecting two entity nodes, as shown in the table. The suffix ":BACK" signals that the second argument of the entity pair comes before the first in the sentence. The trigger is usually taken as the root of the path connecting the two nodes.

For the relations, the columns consist of relation types as defined by a schema. The TAC KBP slot names, such as `per:spouse` or `org:founded_by`, appear as columns with observed cells for entity pairs that are annotated either in TAC KBP training documents or in the reference KB. These columns are crucial for learning from surface patterns, and for predicting a TAC KBP relation between any entity pair. For cold

Feature Type	Example
Dep-path	↑pobj↑in↑prep↑die↓nsubj↓
Words	die in:BACK
Type-Words-Type	ORG-die in:BACK-PER
Type-Trigger-Type	ORG|die|PER

Table 1: Surface patterns used in universal schema. These patterns are extracted for entity pair (Abbey Church, Father Daniel) from sentence "Father Daniel died in the Abbey Church at Saint Anselm"

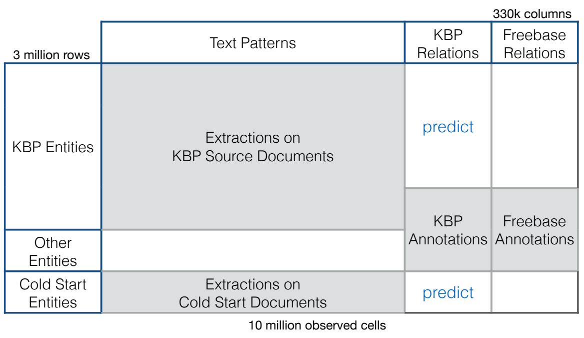


Figure 3: **Universal Schema for for KBP:** An illustration of the complete matrix used for universal schema. The gray areas are partially observed cells, where "KBP Annotations" represent both the KBP reference KB and the training annotations. "Predict" denotes that parts of the matrix that we need to predict. Note that for cold start we do not observe any KBP or Freebase relations.

start, we also include a number of additional entity pairs from Freebase, including columns that denote Freebase relations between them (allowing our system to learn the associations between surface patterns, TAC relations, and the Freebase relations). The data sources used for universal schema for slot-filling and cold-start is illustrated in Figure 3, showing the various observed cells, and the cells that need to be predicted for the tasks.

3.3 Provenance Prediction

Since universal schema models the relations at a corpus-level, the extractions are predicted using a combination of evidence from multiple observed patterns and freebase annotations, making the task of identifying a single provenance of a predicted slot-value non-trivial. Fortunately, the embedding of all relations, both from the target schema and from surface patterns, into a common euclidean space allows us to estimate the most likely provenance for our predictions. First, we compute the pairwise similarities between the embeddings for every pair of columns. For a given prediction, for example X is related to Y by the relation `employee_of`, we consider all surface pattern columns that were observed for the pair (X, Y), and select the pattern that has an embedding most similar to the embedding for `employee_of`. We use the sentence that produced the observed cell for this column as the provenance.

4 Attribute Relation Extraction

Some slots fillers are string-valued (or attribute-value) as opposed to entity-valued. We manually build relation extractors for each attribute slot based on a set of rules. Each extractor takes an entity mention and the sentences in which it appears as input, and outputs a sequence of attribute relation instances. Extractors also provide a confidence score for each extracted relation that is hard-coded and determined empirically. These confidence scores are used in post-processing for determining which relations should be included in the final output. Further, we do not extract all relations, ignoring ones have been historically infrequent such as `political_religious_affiliation`, `number_of_employees`, and `org:website`.

4.1 Trigger Based Extraction

Most of our attribute relation extractors use manually crafted lists of *trigger words* as signals. When an input sentence containing a query mention also contains a trigger word, the corresponding extractor attempts to extract an attribute involving the mention and trigger word. This extractor considers a number of features of the trigger/mention pair such as the number of words separating the two, the word sequence occurring between the mention and trigger, and the order of the mention and trigger. Inspired by recent work (Roth et al., 2012), trigger words for slots `per:cause_of_death`, `per:religion`, and `per:charges` are extracted from respective Freebase categories, while trigger words for other slots consist of small lists (less than 10) of manually-collected words. For example, trigger words for the `per:title` relation include: "secretary," "technician," "'congressman," "rep.," etc.

4.2 Title Extractor

Since titles have been historically extremely prominent in the KBP Slot Filling track, we design additional rules for extracting titles. These rules are based on the dependency parse of the sentence containing a query mention. Specifically, if the parent or a sibling of the query mention in the dependency tree is labeled as an “appositive”, we attempt to extract a title relation between the token and the mention using the type of dependencies.

4.3 Date Extraction

A number of attribute relations require extraction of dates from text, for example `age`, `date_of_birth`, `date_founded`, etc. We identify dates in each document using the Natty date parser (Stelmach, 2013). Extractors for date-related slots require both the presence of a trigger word and a date in the input sentence. Some date-related relation extractors may also extract values of multiple relations. For example, using the publishing date of a document in addition to an assertion of the age of a recently deceased person, the extractor of `age` would also infer the `date_of_death`.

4.4 Alternate Names Extraction

Within-document and cross-document coreference links all the entities in our knowledge base to their mentions in the text corpus. Using this extracted knowledge base, we are able to directly extract `per:alternate_names` and `org:alternate_names` by aggregating the text mentions of all the entities.

5 Relation Resolution

In part due to our slot filling (relation extraction) strategy and in part due to some relation extraction not being performed jointly (e.g. attribute relations), the set of extracted fillers for a mention/slot pair can be invalid with respect to the slot. For example, our relation extraction system could extract inconsistent `per:date_of_birth` values for the same entity. Additionally, our system may extract multiple values for a uniqe-valued relation (for example `per:city_of_birth`. To remedy these problem, we employ a suite of simple `resolvers` that handle these issues on a case by case basis.

For single-valued slots (i.e. `per:age`), we use a resolver that outputs the filler with the highest confidence. For multiple-valued slots (i.e. `per:alternate_names`), we threshold the number of (unique) fillers output by only picking the top k–ranked either by confidence or by the number of appearances (i.e. the filler “50” was extracted for entity `e_1` and relations `per:age` 20 times). By thresholding, we hoped to increase precision without substantially decreasing the recall.

6 Results

In this section we investigate the performance of our system on Slot-Filling and Cold-Start tasks, and analyze the errors made by the system.

6.1 Universal Schema

We use universal schema to learn dependencies between surface patterns and relations, which are used to infer relations between observed entity pairs. Here we provide both the evaluation in terms of the accuracy of the predictions, but also qualitative exploration of the dependencies learned by the approach.

6.1.1 Embeddings

To interpret the embeddings of the surface patterns, we calculate the cosine similarity between vectors of a TAC relation column and other columns. We list the top ranked patterns with respect to each target slot in Table 2. For simplicity, we translate the dependency path to word sequence. To generalize the patterns, we replace tokens of part-of-speech tag “NNP" with their tags. For example, in pattern “replace NNP NNP as face of," those two “NNP" tokens may stand for a person name. The suffix “:INV" indicates that the two arguments are in inverse position. Some patterns are augmented with entity types of the two arguments. Slots that are similar to the target slots are also shown, for example, `per:organizations_founded` is indicative of slot `employee_of`.

We observe here that our approach can learn diverse and accurate patterns that are indicative of the target slots. For example, we extract patterns that contain “ex-wife," “hubby," “file divorce" for `spouse`, patterns that include “pay by," “resign from" for `employee_of`. We also analyze errors made by our approach. Some errors are caused by incorrect dependency path extracted from the text. For example, pattern “X into a career to Y" is closely related to `spouse` due to incorrectly extracting the path for entity pair (Lucinda, Robert Morgenthau) from the sentence “Lucinda into a

Slot	Patterns
employeeOf	PER|pay by state of|LOC, pay by state of, who resign from, have resign in protest from, have be select as candidate of, PER|cartoonist at|UKN, be announce as be, LOC|NNP attend summit of|ORG have make his/her legend since sign from, make start in, work as register lobbyist for, PER|defender in NNP|ORG PER|executive who run|UKN, replace NNP NNP as face of, per:organizations_founded , PER|be play position for|ORG
cityOf Residence	be perform with NNP NNP at, PER|revenue NNP|PER, PER|have be from|ORG PER|be live in|LOC, NNPS on death row in, PER|citizen of|LOC PER|family live in|LOC, PER|his/her brother live in|LOC, be wheel into court in, PER|where X grow up|LOC, who be away from, be son of NNP NNP NNP of, PER|who be a native of|LOC X, a Y prisoner; bounce between his/her home in, since X live in Y
schools Attended	from his/her days in college at, be junior at, revel in NNP NNP success at hockey player for, be graduate of NNP NNP NNP, have be select as candidate of, have resign in protest from, per:employee_or_member_of director of NNP for NNPS at; PER|pay by state of|LOC; X, Y deputy managing director; PER|lawyer be hire by|ORG; policy adviser on NNP at PER|be coach at|ORG, PER|professor of education at|ORG, who earn at
spouse	into a career to, doled punishment, PER|X's ex-husband, Y|PER have file for divorce from her husband, file for divorce from, PER|X, Y's ex-husband|PER, PER|hubby|PER, marry actress, file for divorce from music producer, marry socialite; accompany by his wife; X, Y's ex-wife; X join her/his fiance Y, photographed with fiance, PER|X, Y's girlfriend|PER
subsidiaries	ORG|research growth rate for|LOC, its parent firm NNP NNP NNP, be subsidiary of, X, operated by Y:INV, UKN|populace be obliterate|ORG ORG|headquarters that NNP|ORG, UKN|court while|ORG, headquarters that NNP, sell NNP brokerage business, ORG|mecca|ORG ORG|X, a venture of Y:INV|ORG, ORG|X company Y|ORG, PER|be split into|ORG, ORG|which is bought by:INV|ORG ORG|which own percent of|ORG, ORG|NNP parent of|ORG ORG|parent company of|ORG, own NNP broadcast network
headquarters	ORG|lender base in|UKN, NNP announce at ORG|NNP NNP program in|LOC, ORG|be headquartered in|LOC ORG|think tank in|LOC, ORG|research center in|LOC ORG|X, a Y-based bank|LOC , ORG|X, Y organization|LOC ORG|policy group in|LOC, tenant right organization base in downtown

Table 2: Top similar patterns to the target slots.

Slot	Prec	Rec	F1
children	0.154	0.266	0.195
cities_of_residence	0.238	0.294	0.263
city_of_birth	0.083	0.077	0.080
city_of_death	0.478	0.314	0.379
countries_of_residence	0.149	0.204	0.172
employeeOf	0.435	0.118	0.186
origin	0.020	0.016	0.018
parents	0.023	0.103	0.038
schools_attended	0.063	0.034	0.044
siblings	0.025	0.231	0.044
spouse	0.138	0.271	0.183
state_of_death	0.188	0.120	0.146
states_of_residence	0.109	0.207	0.413
city_of_headquarters	0.125	0.125	0.125
country_of_headquarters	0.160	0.082	0.108
founded_by	0.025	0.100	0.040
member_of	0.007	0.250	0.013
parents	0.036	0.286	0.063
shareholders	0.037	0.412	0.068
members	0.010	0.045	0.017
state_of_headquarters	0.350	0.318	0.333
subsidiaries	0.053	0.235	0.086
top_employees	0.398	0.279	0.328
Overall	**0.094**	**0.183**	**0.125**

Table 3: **Entity-based Slots:** Performance on different slots. We first list person slots, followed by organization slots. Our approach obtains high recall.

Pulitzer Prize-winning career in journalism and marriage to New York District Attorney Robert Morgenthau." Since annotation data is limited, this pattern is in the top ranked list for predicting spouse. Other errors arise since our model is unable to distinguish two slots from each other. For example, some surface patterns that are similar to employee_of are also in the top ranked list for schools_attended. However, our model does learn patterns specific to schools, such as "be junior at," "his/her college in," and "graduate of."

6.1.2 Evaluation of the Predictions

We evaluate the predictions of the universal schema model on the 2013 Slot-Filling task. Table 3 lists precision, recall and F1 measures for different slots. We can see that our approach achieves relatively high recall. As we describe in the previous section, most of our errors arise from relations that can be expressed in significantly different ways (for e.g. member_of), but do not have ample observed patterns.

6.2 Attribute Extraction

We evaluate the attribute extraction system on the 2013 Slot Filling attribute-valued slots in Table 4. As expected, the rule based extractors achieve high precision, while their recall is relatively lower. The better performance of string-valued slots as compared to entity-valued may be explained by the fact that they are likely to be easier to extract than entity-valued relations, as the latter requires combination of information about the entity across multiple documents.

Slot	Prec	Rec	F1
alternate_names	0.150	0.097	0.118
title	0.259	0.199	0.225
charges	0.130	0.058	0.080
date_of_birth	0.400	0.060	0.105
date_of_death	0.290	0.100	0.149
age	0.786	0.224	0.349
cause_of_death	0.417	0.182	0.253
religion	0.200	0.250	0.222
alternate_names	0.234	0.202	0.217
date_founded	0.556	0.313	0.400
Overall	**0.272**	**0.150**	**0.193**

Table 4: **String-based Slots:** Performance on attribute slots of people and organizations. We employ rule based extraction for these slots.

Systems	Precison	Recall	F1
Top-1	42.53	33.17	37.27
Top-10	30.34	9.67	14.67
UMass IESL	**10.88**	**18.46**	**13.69**

Table 5: **KBP 2013 Slot Filling:** Comparison of our system to other participants. We submitted only a single run, are were placed 11th out of the 18 participant teams.

6.3 Comparison to KBP 2013 Participants

We compare our system to the other KBP 2013 participants according to the official evaluation. For the slot-filling tasks, as shown in Table 5, we obtain an overall F1 of 13.7%, with a relatively high recall compared to participants that performed similarly. In Cold-Start, we obtain a combined 0-hop and 1-hop F1 of 7.03%. For the 1-hop evaluation, for which entity-valued relations are more important, our score is higher than NYU's. This perhaps indicates that the our string-valued slot extraction rules perform relatively poor, and this has a significant impact on the overall evaluation.

Systems	0-hop			1-hop			Combined		
	Prec	**Rec**	**F1**	**Prec**	**Rec**	**F1**	**Prec**	**Rec**	**F1**
UMass IESL	25.24	5.06	8.44	11.42	3.46	5.32	17.88	4.38	7.03
NYU	45.81	7.28	12.56	16.35	1.96	3.5	35.15	4.99	8.74
HLT-COE	40.54	32.7	36.2	21.36	11.02	14.54	34.3	23.39	27.81

Table 6: **KBP 2013 Cold Start:** Comparison of our systems to other two participants.

7 Conclusion and Future Work

We presented universal schema, our overall framework for TAC KBP slot filling and cold start. It differs from existing systems in that it considers all entities, relations, surface patterns, and annotations jointly in a holistic manner. We effectively construct a database about the entities that appear in the input documents (training and test), and query this database to provide our submission.

A number of avenues exist for future work. We feel our prediction suffered due to the use of a simple cross-document system and the fact that our within-document coreference did not utilize external information sources. We look forward to incorporating sophisticated hierarchical cross-document coreference (Singh et al., 2011) and KB-supervised within-document coreference (Zheng et al., 2013). Future work will also explore expansion of the features used in the universal schema model, including lexicons, additional Freebase relations, and inferred entity types (Yao et al., 2013).

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by DARPA under agreement number FA8750-13-2-0020. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Eric Bengtson and Dan Roth. 2008. Understanding the value of features for coreference resolution. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 294–303.

Marc-Allen Cartright, Samuel Huston, and H Field. 2012. Galago: A modular distributed processing and retrieval system. In *SIGIR 2012 Workshop on Open Source Information Retrieval*, pages 25–31.

Michael Collins, Sanjoy Dasgupta, and Robert E. Schapire. 2001. A generalization of principal component analysis to the exponential family. In *Neural Information Processing Systems (NIPS)*.

Laura Dietz and Jeffrey Dalton. 2012. Across-document neighborhood expansion: Umass at tac kbp 2012 entity linking. In *Text Analysis Conference (TAC)*.

Fang Liu and Jun Zhao. 2012. Sweat2012: Pattern based english slot filling system for knowledge base population at tac 2012. In *Text Analysis Conference (TAC)*.

A. McCallum, K. Schultz, and S. Singh. 2009. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems Conference (NIPS)*.

Bonan Min, Xiang Li, Ralph Grishman, and Ang Sun. 2012. New york university 2012 system for kbp slot filling. In *Text Analysis Conference (TAC)*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

B. Roth, G. Chrupala, M. Wiegand, M. Singh, and D. Klakow. 2012. Generalizing from freebase and patterns using distant supervision for slot filling. In *Text Analysis Conference (TAC)*.

Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. 2011. Large-scale cross-document coreference using distributed inference and hierarchical models. In *Association for Computational Linguistics: Human Language Technologies (ACL HLT)*.

Joseph Stelmach. 2013. Natty, October.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Third Workshop on Automated Knowledge Base Construction*, October.

Jiaping Zheng, Luke Vilnis, Sameer Singh, Jinho Choi, and Andrew McCallum. 2013. Dynamic knowledge-base alignment for coreference resolution. In *Conference on Computational Natural Language Learning (CoNLL)*.