New York University 2011 System for KBP Slot Filling

Ang Sun, Ralph Grishman, Wei Xu, Bonan Min
Computer Science Department
New York University
{asun, grishman, xuwei, min}@cs.nyu.edu

1 Introduction

The NYU Knowledge Base Population (KBP) slot filling system for 2011 was built upon the system for 2010. The primary addition was a set of classifiers trained using distant supervision. A secondary addition involved a slot-specific passage retrieval system combined with name-type-based answer selection -- in effect, a simple QA system. In the three sections that follow, we review the 2010 baseline system and describe the distant supervision and QA components. Following that, we report on this year's results, including the contribution of the individual system components.

2 Baseline: NYU KBP 2010 system

We describe first the baseline system, whose basic structure was unchanged from 2010 (Grishman and Min 2010), although a number of small refinements were made based on an error analysis of last year's results. These included a larger list of titles and an improved rule for distinguishing members from employees.

The NYU system, like most KBP systems, has 3 basic components: document retrieval, answer extraction, and merging.

Document retrieval uses Lucene to retrieve a maximum of 300 documents from the corpus; the retrieval query consists of the query name and some minor name variants (omitting middle initials, corporate suffixes). To avoid bogging down the system with the occasional very long document, documents exceeding 40,000 characters are ignored. No use is made of the document given in the query to disambiguate ambiguous names.

Answer extraction begins with text analysis – part-of-speech tagging, chunking, name tagging, time expression tagging, and coreference – performed using the NYU Jet system. The results of coreference are used to fill the KBP alternate_names slots. Other slots are filled through a combination of hand-coded patterns and patterns created semi-automatically using bootstrapping. These patterns are organized into a set of response generators which are applied independently to the document; after all generators have been applied, we select the best answer (for single-valued slots) or eliminate redundant answers (for list-valued slots).

The hand-coded patterns operate within a noun group or between two noun groups connected by a preposition. These include the pattern sets in Table 1.

A few notes:

1. GPE = geo-political entity, a location with a government [an ACE semantic class]
2. Titles are recognized using a list of approximately 600 titles gathered from Wikipedia infoboxes and edited by hand. Titles preceded by 'assistant', 'deputy', or 'vice' are excluded for the top_members/employees slot.
3. Employee_of is distinguished from member_of based on the type of organization: non-governmental and sports organizations (based on the ACE classification) have members; other organizations have employees. As a special case, coaches and managers of sports organizations are treated as employees.

pattern set	patterns	slots
local patterns for *person* queries	title of org, org title, org's title, title	title, employee_of
	title in GPE, GPE title	origin, location_of_residence
	person, integer,	age
local patterns for org queries	title of org, org title, org's title	top_members/employees
	GPE's org, GPE-based org, org of GPE, org in GPE	location_of_headquarters
	org's org	subsidiaries / parent
implicit organzation	title [where there is a unique org mentioned in the current + prior sentence]	employee_of [for person queries]; top_members/employees [for org queries]
functional noun	F of X, X's F where F is a functional noun	family relations; org parents and subsidiaries

Table 1: Pattern Sets

4. Location_of slots are divided into countries, states, and cities based on simple table look-up, using tables of counties and U.S. states (anything not in this table is classified as a city)

The bootstrapped patterns are created starting from a small set of seed patterns for each KBP slot, growing the set through semi-supervised learning using the KBP corpus, and manually reviewing the resulting patterns. Separate sets of patterns are produced matching linear token sequences and matching dependency tree paths. The dependency structures are built using the Stanford English parser. The bootstrapping process is described in detail in last year's report (Grishman and Min 2010).

Post-processors are applied following pattern matching: to distinguish employees from members, and to distinguish countries, states, and cities.

Finally the merging stage combines answers from different documents and passages, and from different answer extraction procedures.

3 Distant supervision

The primary addition to our KBP system was a set of maximum entropy classifiers for slot filling, trained through distant supervision (Mintz et al. 2009). The training procedure used the Freebase knowledge base and the KBP corpus. Selected relations in Freebase were mapped to 29 slots in the KBP task. The mapping table used was similar to the one in Chen et al. (2010). Roughly 4.1 million relation instances were used for training (See Table 2 for more details). Given a pair of names $< i, j >$ occurring together in a sentence in the KBP corpus, we treat it as a positive example if $< i, j >$ is a Freebase relation instance and as a negative example if $< i, j >$ is not a Freebase instance but $< i, j' >$ is an instance for some j'≠j. These examples are used for both training maximum entropy classifiers and computing the precision of dependency patterns extracted from them. High-precision patterns based on the precision measure proposed in Section 3.2 for a few slots[1] were reviewed by hand, producing a set of 792 patterns for filling slots using strict pattern matching. The resulting classifiers and the pattern matcher were used as additional sources of slot fills, generated in parallel with pattern matching from last year, and merged in the final stage of processing. This produced a significant improvement in overall performance (see Table 5), but not as much as we hoped.

Below we first describe the features used for training classifiers and then discuss a few decisions that we made during the development of our distant learner: refinement of distantly generated

[1] Due to time limitations, we only did this for the four slots: per:employee_of, per:member_of, org:founded_by, org:top_members/employees

labels to reduce noise, undersampling of the majority class to achieve a more balanced class distribution and the use of coreference to incorporate more available information of the query entity for slot filling.

Slot	# Instances
org:alternate_names	1049
org:dissolved	2934
org:founded	57988
org:founded_by	8203
org:country_of_headquarters	248315
org:stateorprovince_of_headquarters	
org:city_of_headquarters	
org:political/religious_affiliation	1735
org:top_members/employees	67244
per:cause_of_death	53149
per:children	33042
per:employee_of	97167
per:member_of	256573
per:origin	466211
per:parents	33043
per:country_of_birth	403519
per:stateorprovince_of_birth	
per:city_of_birth	
per:country_of_death	109182
per:stateorprovince_of_death	
per:city_of_death	
per:religion	80053
per:countries_of_residence	170562
per:stateorprovinces_of_residence	
per:cities_of_residence	
per:schools_attended	162737
per:siblings	8211
per:spouse	16588
per:title	1832723
Total	4110228

Table 2: Freebase Instances for KBP Slots

3.1 Feature Sets

We extract features from both the token sequence and the dependency tree representations of the sentence containing the two names. We also extract features that capture the string and entity type information of the names, the order of the two names in the sentence (same or reversed relative to their order as a Freebase name pair) and a binary feature indicating whether there is a title in between the two names. Table 3 shows a brief description of the features and samples extracted for the Freebase name pair <*Ray Young, General Motors*> in the sentence "*Ray Young*, the chief financial officer of *General Motors*, said GM could not bail out Delphi". For more detailed descriptions of the features and the intuition of why these features are extracted, please refer to (Sun et al., 2011).

3.2 Class Label Refinement

"All men are created equal", unfortunately, is not true in knowledge bases such as Freebase. For example, some people have employers and residences while others do not. When we perform distant learning, i.e., matching Freebase records against unstructured texts to generate training examples, we typically label an example as negative if it has no corresponding entry in Freebase. However, Freebase is highly incomplete and hence many false negative examples are generated. Examples that match entries in Freebase are not always positive in reality, but are assumed to be and labeled as positive examples by distant learning. For example, the sentence "*Bill Gates* has declared war on *Microsoft*'s insecure software" would be labeled as a positive example for the relation *org:founded_by* although the context does not indicate that relation. Even worse is the pervasive phenomenon where pairs of names are often connected by non-relational contexts such as conjunctions and the punctuation comma. This results in many false positive examples.

To alleviate the impact of false positive and negative examples on the quality of learned models, we refine the class labels of distantly generated examples. Specifically, we extract dependency patterns from these examples and estimate the precision of a pattern for a class based on the statistics of the distantly generated class labels. The precision of a pattern p for the class c_i is defined as the number of occurrences of p in the class c_i divided by the number of occurrences of p in any of the classes c_j:

$$prec(p,c_i) = \frac{count(p,c_i)}{\sum_j count(p,c_j)}$$

Feature		Description	Feature Value
Dependency Tree Features	*dpath*	Shortest path connecting the two names in the dependency parsing tree coupled with entity types of the two names	*PERSON appos officer prep_of ORGANIZATION*
	e1dh	The head word for name one	*said*
	e2dh	The head word for name two	*officer*
	same_e12dh	Whether *e1dh* is the same as *e2dh*	*false*
	e1dw	The dependent word for name one	*officer*
	e2dw	The dependent word for name two	*nil*
Token Sequence Features	*tpattern*	The middle token sequence pattern	*, the chief financial officer of*
	ntw	Number of words between the two names	*6*
	wbf	First word in between	*,*
	wbl	Last word in between	*of*
	wbo	Other words in between	*{the, chief, financial, officer}*
	bm1f	First word before the first name	*nil*
	bm1l	Second word before the first name	*nil*
	am2f	First word after the second name	*,*
	am2l	Second word after the second name	*said*
Entity Features	*e1*	String of name one	*Ray_Young*
	e2	String of name two	*General_Motors*
	e12	Conjunction of *e1* and *e2*	*Ray_Young--General_Motors*
	et1	Entity type of name one	*PERSON*
	et2	Entity type of name two	*ORGANIZATION*
	et12	Conjunction of *et1* and *et2*	*PERSON--ORGANIZATION*
Semantic Feature	*mTitle*	Title in between	*true*
Order Feature	*order*	1 if name one comes before name two; 2 otherwise.	*1*

Table 3: Feature Sets

The class label refinement algorithm utilizes two types of precision, *prec1* and *prec2*, computed with the sum including and excluding the class *OTHER* (not a defined KBP slot), respectively.

Let
top_class1(p) = the class c (including *OTHER*) which maximizes *prec1*(p, c)
top_class2(p) = the class c (excluding *OTHER*) which maximizes *prec2*(p, c)
top_prec1(p) = the max of *prec1*(p, c)
top_prec2(p) = the max of *prec2*(p, c)

Then we refine the class of p using the following rule:

```
if (top_class1(p) == "other") {
   if (top_score2(p) < 0.5 | top_score2(p) == 1)
      return "other"
      else return top_class2(p)}
else if top_score1(p) >= 0.3
   return top_class1(p)
   else return "other"
```

Cutoff values were selected by eyeballing the quality of patterns.

To see why this refinement gives more accurate class labels, taking the pattern "*appos* chairman *prep_of*" as an example; most of the time it was labeled as *OTHER* because of the incompleteness of Freebase. After we compute *prec1* and *prec2*, every example (including those were labeled as *OTHER*) that is associated with this pattern will be refined to the class *per:employee_of*.

Figures 1, 2 and 3 show the results averaged on 10 runs on the 2011 evaluation data using different undersampling ratios, which is defined as the ratio between negative and positive examples (see Section 3.3 for more details). Under each setting of the undersampling ratio, models trained with refined class labels (*MR* and *SR*) outperformed models trained without refined labels (*MNR* and *SNR*) by large margins. This indicates the superiority of the proposed class label refinement method in distant learning for slot filling. It is also noticeable that the performance curves of models trained with refined labels are much flatter than those of models trained without refined labels, reflecting that they are less sensitive to the undersampling ratio parameter and more robust to noise.

Slot	Freq.	prec1	prec2
per:member_of	59	0.038	0.175
org:top_members /employees	20	0.013	0.059
per:employee_of	254	0.164	**0.754**
org:founded_by	4	0.003	0.012
OTHER	1208	**0.782**	NA

Table 4: Pattern "*appos* chairman *prep_of*"

3.3 Undersampling the Majorities

Distant learning also produces an extremely unbalanced class distribution. Traditionally, a single n-way classifier is trained to distinguish among the *n* classes. We empirically found that this classifier is biased towards a few majority classes and tends to ignore the minority classes. For example, the single n-way classifier on average produced 180 fills for only 8 slots. As an alternative, we train one n-way classifier for each pair of named entity types. For example, we train a multi-class classifier for the entity type pair PERSON and ORGANIZATION and train another one for PERSON and PERSON. There is still a majority class for each such n-way classifier, namely the class that cannot be mapped to any KBP slot even after label refinement. We then downsize the majority class by randomly selecting a subset of its examples.

Multiple n-way classifiers not only produced more fills for more slots (on average 240 fills for 15 slots), but also provided better F-Measure than the single n-way classifier. This was mainly achieved by an improved recall as shown in Figure 3.

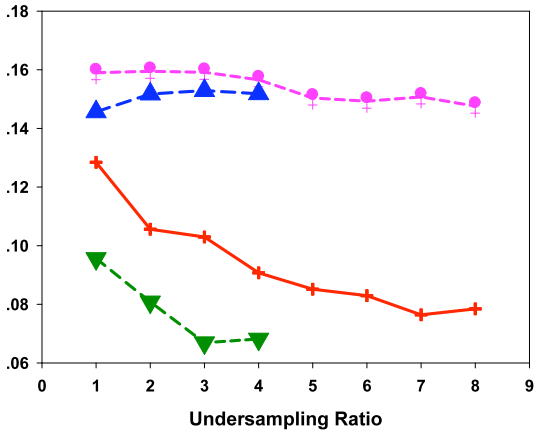

Figure 1: F-Measure

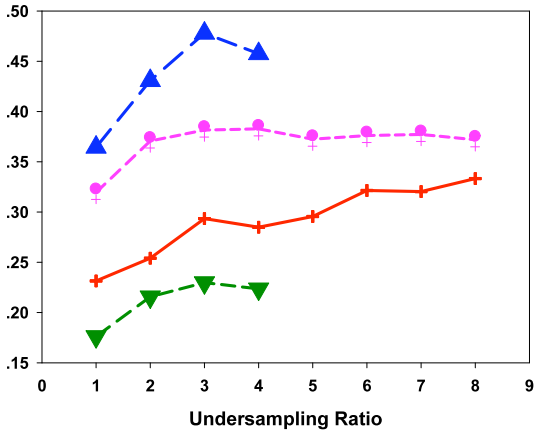

Figure 2: Precision

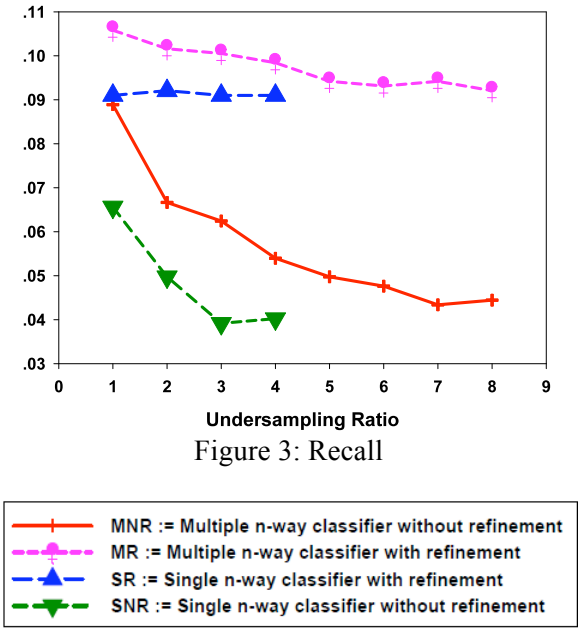

Figure 3: Recall

3.4 Contribution of Coreference

For the sake of matching accuracy, the training of distant learning relies on strict match of names. When we actually fill slots for a given query, its co-referred names in a single document can be provided by a co-reference module. Our submitted runs used the co-referred names of the query in both the distantly trained classifiers and the simple dependency pattern matcher which together achieved P/R/F of 36.4/11.4/17.4 on the 2011 evaluation data. The use of co-reference is clearly beneficial to our system; without using it the P/R/F dropped to 28.8/10.0/14.3. Note that the results in the three figures above were also obtained using the coreference information.

4 Passage retrieval / QA

One further component incorporated into this year's system was based on passage retrieval and name typing. For each slot, a set of index terms is generated using distant supervision (again, using Freebase) and these terms are used to retrieve and rank passages for a specific slot (Xu et al. 2011). An answer is then selected based on name type and distance from the query name. This is used as a fall-back strategy, if the other answer extraction components did not find any slot fill. Due to limitations of time, this procedure was only implemented for a few slots (employer and headquarters location) and the constraints were not tight enough to improve overall slot-filling performance.

module	score using only module			score excluding module		
	recall	precision	F1	recall	precision	F1
distant supervision overall	11.4	36.4	17.4	20.2	35.4	25.7
distant supervision classifier	10.0	37.9	15.4	21.1	34.5	26.2
distant supervision pattern matcher	2.0	28.6	3.6	24.7	35.7	29.2
alternate names	2.8	45.7	5.4	23.0	34.1	27.5
local patterns	14.4	41.0	21.3	18.5	33.4	23.8
implicit organization	0.6	5.5	1.1	25.0	39.2	30.5
functional nouns	0.5	23.8	1.0	25.1	35.3	29.3
bootstrapped linear patterns	3.5	54.1	6.6	24.8	34.6	28.9
bootstrapped dependency patterns	1.8	36.2	3.4	25.0	35.2	29.2

Table 5: Ablation Study of the System NYU2

5 Results

Because the passage/QA component was added at the last minute, we thought it prudent to submit one run with this component (NYU1) and one run without (NYU2). It turned out (as just noted) that this component was underconstrained. It added 35 slots fills, only two of which were correct; NYU2 had better performance:

	Recall	Precision	F1
NYU1	25.7	33.6	29.1
NYU2	25.5	35.0	29.5

Table 6: Performance of NYU Systems

Using NYU2 as a base, we then measured the contribution of each module in isolation and the performance of the system when each module was removed in turn (ablation study).[3] As one can see from Table 5, the hand-coded local patterns by themselves and the classifier trained by distant supervision by itself provided quite good performance. There is a lot of overlap between the contributions of the different modules, but the ablation study indicates that all modules made a positive contribution to the final result except for the implicit organization module, which had satisfactory precision on the training data but very poor precision on the test data.

Acknowledgments

Supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory (AFRL) contract number FA8650-10-C-7058. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL, or the U.S. Government.

[3] Because the results of multiple modules are merged at the end, removing one module may lead to the generation of a new result (produced by a different module) – a result which has not been assessed. Such responses are scored as incorrect by the scoring program, though some may in fact be correct. In consequence, the numbers in this table somewhat understate the actual results.

References

Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Wen-Pin Lin, Javier Artiles, Matthew Snover, Marissa Passantino and Heng Ji. 2010. CUNY-BLENDER TAC-KBP2010 Entity Linking and Slot Filling System Description. *Proc. Text Analytics Conference (TAC2010).*

Ralph Grishman and Bonan Min. New York University KBP 2010 Slot Filling System. *Proceedings of Text Analysis Conference 2010.*

Mike Mintz, Steven Bills, Rion Snow and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. *Proceedings of ACL-IJCNLP 2009.*

Ang Sun, Ralph Grishman and Satoshi Sekine. Semi-supervised relation extraction with large-scale word clustering. *Proceedings of ACL 2011*.

Wei Xu, Ralph Grishman and Le Zhao. 2011. Passage retrieval for information extraction using distant supervision. *Proceedings of IJCNLP 2011*.