Top-down and Bottom-up: A Combined Approach to Slot Filling

Zheng Chen, Suzanne Tamang, Adam Lee, Xiang Li, Marissa Passantino and Heng Ji

Computer Science Department, Queens College and Graduate Center
City University of New York, New York, NY 13367, USA
hengji@cs.qc.cuny.edu

Abstract. The Slot Filling task requires a system to automatically distill information from a large document collection and return answers for a query entity with specified attributes ('slots'), and use them to expand the Wikipedia infoboxes. We describe two bottom-up Information Extraction style pipelines and a top-down Question Answering style pipeline to address this task. We propose several novel approaches to enhance these pipelines, including statistical answer re-ranking and Markov Logic Networks based cross-slot reasoning. We demonstrate that our system achieves state-of-the-art performance, with 3.1% higher precision and 2.6% higher recall compared with the best system in the KBP2009 evaluation.

Keywords: Slot Filling, Information Extraction, Question Answering

1 Introduction

The increasing number of open evaluations and shared resources has made it possible for many natural language processing tasks to benefit from system combination. Fortunately, the Slot Filling track of the recently launched Knowledge Base Population (KBP) task[1] at the Text Analysis Conference (TAC) provides a platform to attempt system combination for information extraction.

The KBP Slot Filling task involves learning a pre-defined set of attributes for person and organization entities based on a source collection of documents. A query contains a name-string, docid, entity-type, node-id (entry ID) in Wikipedia, an optional list of slots to ignore. For example, *[Andy Warhol, ABC-20080611-9372, PER, SF7, per:date_of_birth]* is a query for the painter "*Andy Warhol*", for which a system should return all slot types except *per:date_of_birth*. KBP 2010 defined 26 slot types for persons and 16 slot types for organizations. This task has attracted many participants from IE and Question Answering (QA) communities. In addition, a large amount of system or human annotated data are shared as a community effort.

In this paper we present a state-of-the-art Slot Filling system that includes two bottom-up IE style pipelines and a QA style pipeline, with several novel enhancements including statistical answer re-ranking and Markov Logic Networks (MLN) based cross-slot reasoning. We evaluate performance across our pipelines, with the systems from KBP2009 and human annotators.

[1] http://nlp.cs.qc.cuny.edu/kbp/2010/

2 System Overview

Figure 1 depicts the general procedure of our approach. All three pipelines begin with an initial query processing stage where query expansion techniques are used to improve recall. The next step of the system is pipeline dependent, representing three alternative approaches to the KBP task: IE, pattern matching and QA. After generation of the best answer candidate sets form the individual systems, they are combined to re-rank confidence on the system-wide answer set and for cross-slot reasoning.

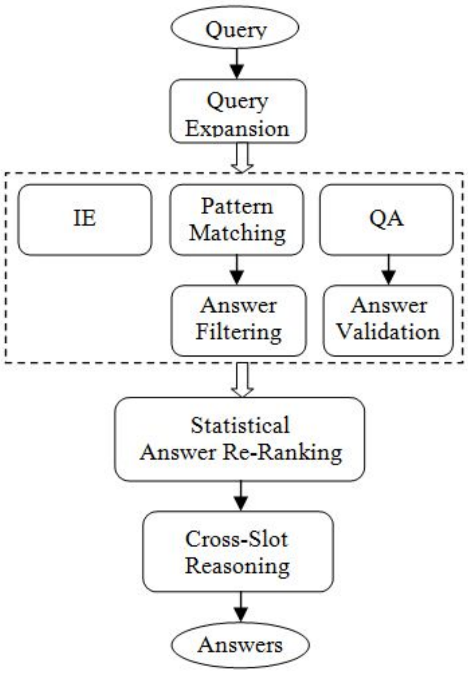

Figure 1. Slot Filling System Pipelines

3 Bottom-up and Top-down Pipelines

These pipelines are organized in two forms: bottom-up IE based approaches that extract all possible attributes for a given query and then fill in the slots by mapping and inference (section 3.1 and 3.2); and top-down QA based approach that search for answers constructed from target entities and slot types (section 3.3).

3.1 Pattern Matching

In pattern matching approach, we first automatically obtain the ranked patterns by learning from *query-answer (q-a)* pairs, and then apply these patterns to find answers to unseen queries. For example, given the pair *(Michael Jackson, 50)* for slot *per:age*, we can extract sentences in which Michael Jackson and 50 co-occur:

(1) Michael Jackson died at the age of 50 ; (2) Michael Jackson (50) …

Pattern can be constructed as

(1) *<Q>* died at the age of *<A> ;* (2) *<Q> (<A>)*

This approach consists of the following steps:

(1) Selection of query-answer pairs

We extract *q-a* pairs from the facts listed in Wikipedia infobox[2] by some mappings from infobox fields to KBP slots. *q-a* pairs are split into two sets: half for pattern extraction, and the other half for pattern assessment.

(2) Pattern extraction

For each *q-a* pair from the training set, we use a search engine[3] to retrieve the top 1000 documents in the source collection, and pick out sentences in which the query and answer co-occur. In addition to populating static patterns for different *q-a* pairs, we also apply entity type replacement and regular expressions to make patterns as general as possible.

(3) Pattern assessment

For each *q-a* pair from a stand-alone development set, we search the top 1000 documents and pick out the sentences in which the query occurs. We apply patterns for each sentence and if it can be matched, extract the entity at the exact place as the answer. We sort the patterns in the descending order of precision (matching rate), and filter those with precision below a threshold.

(4) Pattern matching

To obtain candidate answers for slot filling, we locate the sentences where *q* occurs, apply the patterns generated by step (3) and extract the answer when there is a pattern match. We then rank the answers according to the sum of precisions of all patterns that produce the answer.

(5) Filtering answers

We set a low threshold to include more candidate answers, and then apply several filtering steps to distill the best answers. Filtering steps include removing answers with inappropriate entity types, erroneous answers that are not in dictionary resources (e.g., the country dictionary for slot *per:country_of_birth*) or inappropriate answers whose dependency parsing paths to the query do not satisfy certain constraints (e.g., for slot o*rg:subsidiaries, org:parents*, the query and the answer should not have a conjunction relation).

3.2 Supervised IE

We apply a cross-document English IE system (Ji et al., 2009) to extract relations and events defined in NIST Automatic Content Extraction Program (ACE 2005)[4]. Relation extraction and event extraction are based on maximum entropy models, incorporating diverse lexical, syntactic, semantic and ontological knowledge. ACE2005

[2] http://en.wikipedia.org/wiki/Help:Infobox
[3] http://lucene.apache.org/
[4] http://www.nist.gov/speech/tests/ace/

defines 6 main relation types and 18 subtypes; 8 event types and 33 subtypes. We apply the following mapping between ACE relation/event and KBP.

Given a 3-tuple $<em_i, em_j, r>$ from relation extraction which indicates that the entity mentions em_i and em_j holds a relation r, and if r matches a slot type r' and em_i matches the query entity q in slot filling, then the answer a in the uncompleted 3-tuple $< q, a, r'>$ for slot filling is em_j.

Given a 3-tuple $<t, arg, e>$ and arg = $\{em_i, em_j, \ldots\}$ from event extraction which indicates that the trigger word t indicates an event type e and the involving arguments in *arg* include em_i, em_j, and so on. If the event type e matches a slot type e',em_i matches the query entity q in slot filling and em_j satisfies the role constraint, then the answer a is em_j. For example, if a *MARRY* event matches *per:spouse* slot, and one person entity em_i matches the query, and the other involved entity em_j satisfies the role constrain*t of PER*SON, then we return em_j as the answer.

3.3 Question Answering

We also apply the web module of an open domain QA system, OpenEphyra (Schlaefer et al., 2007) to retrieve candidate answers for the KBP slot filling task. Since candidate answers must be entailed in the KB and a corresponding document id identified, additional answer processing is necessary to determine the candidate answer's relevance and retrieve the corresponding docid(s) for the document collection.

To estimate the relevance, *R*, of a *q-a* pair, we use the joint probability of observing both the query and answer by means of the answer pattern probability:

$$P(q, a) = P(q\ NEAR\ a)$$

where *NEAR* is defined as within the same sentence boundary. At the sentence level, we calculate the frequency of *q-a* pair occurrence in the reference corpus and modify the related Corrected Conditional Probability (CCP) formula to assess *R* for query pattern *q* and answer pattern *a*:

$$R(q,a) = \frac{frequency(qNEARa)}{frequency(q) * frequency(a)} * \#totalsentences$$

After the relevance scores are calculated, the values for each KBP slot are rescaled from 0-1 in order to facilitate the comparison of relevance values among different slot.

4 More Queries and Fewer Answers

We enhance the above three pipelines based on the following extensions. We hypothesize that cleverly designed query expansion techniques (section 4.1) will improve recall of candidate answers to the query. By obtaining more potential correct answers in the ranked list, we can exploit effective learning-to-rank technique to select the best answer (section 4.2). Furthermore, most slot filling methods often produce logically incoherent answers. We design a novel cross-slot reasoning approach based on Markov Logic Networks (MLN) to further refine the quality of answers and predict new answers (section 4.3).

4.1 Query Expansion

In order to generate informative natural language questions from each pair of <query name, slot type>, we develop the following expansion methods.

(1) Template expansion

We generated 59 question templates for the 16 organization slots and 62 question templates for the 26 person slots. For example, the following semantically equivalent questions are generated for the "person: age" slot type:

- What is <per>'s age?
- How old is <per>?
- When was <per> born?

During candidate answer generation, the tag <per> is replaced by the target. On average, each target value produced an initial set of 112 candidates per slot. After filtering by stop words, and sufficient co-occurrence with the query and answer pattern in the reference corpus, queries averages 4.9 answers each in the baseline results, which suggests a very high rate of spurious results from the web module. For this reason, query expansion is a necessary step in the QA pipeline and a rough estimate using a small set of queries without enhanced expansion suggests the impact of this step on recall leads to approximately a four-fold improvement.

(2) Name expansion

The query name may be mentioned in its alternative names in the corpus, thus, name expansion can help improve the recall of slot filling. Wikipedia uses redirect links to indicate navigations among pages that mention the same entity. For example, the entity name "Seyed Ali Khamenei" is redirected to "Ali Khamenei". We mine redirect links from Wikipedia database (dump up to Mar., 2010) and use them as dictionary resources to form extra query names.

4.2 Statistical Re-Ranking

We develop a Maximum Entropy (MaxEnt) based supervised re-ranking model to re-rank candidate answers for the same slot. We train our model from 452 labeled *q-a* pairs to predict the confidence of each candidate answer's correctness. We incorporate the following semantics and global statistics based features into the re-ranker:

- **Web Module Rank.** We use the answer confidence assigned by the OpenEphyra system as a re-ranking feature. For a query, this feature can provide information on the confidence of the answer based on web results.
- **Answer Validation Score.** We calculate the answer relevance *R(q, a)* for each query pair in the answer validation procedure. This feature provides confidence information based on co-occurrence in the document collection.
- **Answer Name Type.** We incorporate the name type of the candidate answer (persons, geo-political, organizations, etc.) as an additional feature for re-ranking. Names are tagged using dictionary files compiled by the Ephyra project.

− **Slot Type.** Using the KBP slot type as a feature allows us to re-rank slot types so that our QA system is more likely to get correct answers for a slot type with a higher confidence.

4.3 MLN based Cross-Slot Reasoning

In the slot filling task, each slot is often dependent on other slots, but systems built for the slot filling task often ignore these dependencies and process each slot individually. In particular, the family slots include such dependency relationships (e.g. *X* is *per:children* of *Y* ➔ *Y* is *per:parents* of *X*; *X* is *per:spouse* of *Y* => *Y* is not likely to be *per:siblings* of *X*). Therefore we develop a reasoning component to approach a real world acceptable answer in which all slot dependencies are satisfied. On the other hand, we can design propagation rules to enhance recall, for example, *X* was born on date *Y* → *X*'s age is approximately (the current year – *Y*).

We noticed that heuristic inferences are highly dependent on the order of applying rules, and the performance may have been limited by the thresholds which may over-fit a small development corpus. We use Markov Logic Networks (Richardson and Domingos, 2006), a statistical relational learning language, to model these inference rules more declaratively. Markov Logic extends first order logic in that it adds a weight to each first order logic formula, allowing for violation of those formulas with some penalty. We use the Alchemy toolkit (Kok et al., 2007) to encode inference rules such as:

SpouseOf(a,b) →*(~ChildOf(a,b)* ∧ *~ParentOf(a,b)* ∧ *~OtherFamilyOf(a,b)* ∧ *~SiblingOf(a,b)*

Then our remaining uncertainty with regard to this formula will be captured by a weight associated with it. Markov Logic will make it possible to compactly specify probability distributions over these complex relational inferences, and easily capture non-deterministic (soft) rules that tend to hold among slots but do not have to. We incorporate hard rules such as name/date/number/title format constraints for slots including *per:title*, *per:country_of_death* and *org:number_of_em ployees/members*, as well as soft rules such as *per:birth_of_date* to *per:age* propagation.

5 Experimental Results

In this section we present the overall performance of our three pipelines by comparing to the best system at KBP2009 evaluation and human annotation, and break down the performance to demonstrate the impact of key techniques.

5.1 Data and Scoring Metric

We randomly select 21 queries (11 persons and 10 organizations) and the entire source collection (1,289,649 documents in total) from KBP 2009 evaluation corpora to evaluate our methods[5]. For each query, we combine the human annotation from LDC based on exhaustive search (110 instances) and the correct answers from KBP 2009 human assessment (195 instances) to form our initial answer keys (***K***) including 263 unique instances.

Despite of the requirement conducting search as exhaustive as possible, a single human annotator can only achieve lower than 50% recall, we follow the human assessment procedure as in KBP 2009 evaluation. We ask another human annotator to manually check all those instances generated by the system but not in human annotation, and if an instance is correct, it will be added to form the expanded answer key set (***K'***). Since this human assessment procedure is time consuming, we only apply it to the final best pipeline for comparison, while use relative scores for the detailed break down analysis experiments.

We follow the KBP 2010 scoring metric[6] to evaluate each pipeline. This is a uniform scoring metric based on standard Precision, Recall and F-measure. Since only non-NIL answers are informative in applications, we focus on scoring non-NIL answers. A unique answer instance <query, slot type, answer> is considered as correct if it matches any instance in the answer key. We added additional answer normalizations to the scorer in order to get more reliable scores and speed up human assessment (normalized 6% instances in the test set). The normalizations are based on a list of 362 country name variants (e.g. "the United States = USA") and a list of 423 person fullname-nickname pairs. If a system returns an answer set ***S***, we compute the following scores:

- Relative Precision = $\#(K \cap S) / \# S$ Relative Recall = $\#(K \cap S) / \# K$
- Precision = $\#(K' \cap S) / \# S$ Recall = $\#(K' \cap S) / \# K'$

5.2 System/Human Comparison

Table 3 presents the relative scores for three single pipelines. Although these scores were evaluated against an incomplete answer key set, it indicates that pattern matching can achieve the best result. Supervised IE method performs the worst because not all of the slot types have corresponding relation and event types. QA method obtained comparable precision but much lower recall because the candidate answers are restricted by query template design and the annotations (e.g. name tagging) used for answer validation.

Table 3. Pipeline Relative Performance against Incomplete Answer Keys (%)

Pipeline	Relative Precision	Relative Recall	Relative F-measure
Supervised IE	13.09	12.82	12.95
Pattern Matching	18.95	18.46	18.70
QA	18.97	11.11	14.02

[5] In order to compare with KBP2009 participants, we mapped some fine-grained KBP 2010 slot types back to KBP2009 slot types (e.g. maped "country_of_birth", "stateorprovince_of_birth" and "city_of_birth" back to "place_of_birth").

[6] http://nlp.cs.qc.cuny.edu/kbp/2010/scoring.html

One advantage of QA is that answer candidates can still receive a high confidence despite the context sentence structure. For example, the QA system is the only approach that returned a correct answer for the slot *org:headquarters*, producing the query answer pair <*Convocation of Anglicans in North America, Nigeria*> and returning the context sentence "*In May, Global South spokeman and Nigerian Archbishop Peter Akinola consecrated Martyn Minns of Virginia as Bishop of the Church of Nigeria for an outreach programme called Convocation of Anglicans in North America.*"

The most common mistakes of the QA pipeline are directly related to the use of co-occurrence statistics. Although semantic evidence indicates an answer is invalid, it is still returned. For example, the QA pipeline identified context sentence, "*Masked fighters parade beneath yellow flags beside the faces of Nasrallah and Abbas Moussawi , Nasrallah's predecessor who was assassinated, along with his wife and son, in an attack by an Israeli helicopter pilot*", but since only co-occurrence were used, the following inaccurate answer pair was generated for the slot *per:title* <*abbas moussawi, assassinated*> and should have been returned for *per:cause_of_death*.

In addition, we compared our absolute scores with the LDC human annotator and the top slot filling system in KBP2009 evaluation which achieved the best score for non-NIL slots. The results are shown in Table 4.

Table 4. Performance Comparison with State-of-the-art System and Human Annotator

System/Human	Precision	Recall	F-Measure
Pattern Matching	**35.26**	**34.36**	**34.81**
09 Best System	32.12	31.79	31.96
Single LDC Human Annotator	100	41.83	58.99

From Table 4 we can see that our pattern matching approach achieved significantly better results than the top site at KBP2009 evaluation, on both precision and recall. We can also conclude that Slot Filling is a very challenging task even for human because the human annotator can only find 41.83% answers.

5.3 Impact of Statistical Re-Ranking

We then evaluate the impact of combination methods and will demonstrate they are also essential steps to achieve good slot filling results. We will describe the results for statistical re-ranker in this subsection and cross-slot reasoning in 5.4 respectively.

We evaluate the impact of re-ranking on the QA pipeline. The relative scores are presented in Table 5. We can see that statistical re-ranking significantly improved (about 5%) both precision and recall of the QA pipeline.

Table 5. Statistical Re-Ranking on QA (%)

QA Pipeline	Relative Precision	Relative Recall	Relative F-measure
Before Re-Ranking	16.41	6.54	9.36
After Re-Ranking	18.97	11.11	14.02

Supervised Re-Ranking helps to mitigate the impact of errors produced by scoring based on co-occurrence. For example, when applied to our answer set, the answer "*Clinton*" for the query "*Dee Dee Myers*", had a system relevance score for the attribute *per:children* due to frequent co-occurrence that was reduced and consequently removed by re-ranking. Alternatively, the query "*Moro National Liberation Front*" and answer "*1976*"did not have a high co-occurrence in the text collection, but was bumped up by the re-ranker based on the slot type feature *org:founded* .

5.4 Impact of Cross-Slot Reasoning

Experimental results demonstrate the cross-slot reasoning approach described in section 4.3 can enhance the quality of slot filling in two aspects: (1) It can generate new results for the slots which the pipelines failed altogether; (2) It can filter out or correct logically incoherent answers. For the test set, this method significantly improved the precision of slot filling without any loss in recall. Table 6 presents the number of spurious errors removed by this approach when it is applied to various pipelines.

Table 6. Impact of Cross-Slot Reasoning

Applied Pipeline		#Removed Errors
QA	Baseline	30
	+Validation	23
	+Re-Ranking	16
Supervised IE		7
Pattern Matching		3

6 Related Work

The task of extracting slots for persons and organizations has attracted researchers from various fields, e.g., relation extraction, question answering, web people search, etc. Most KBP systems follow one of the three pipelines we described, such as IE-based approaches (Bikel et al., 2009) and QA based methods (Li et al., 2009). Some previous work (e.g. Schiffman et al., 2007) demonstrated that IE results can be used to significantly enhance QA performance.

Answer validation and re-ranking has been crucial to enhance QA performance (e.g. Magnini et al., 2002; Peñas et al., 2007). Recent work (Ravichandran et al., 2003) has showed that high performance for QA systems can be achieved using as few as four features in re-ranking. Our results on the QA pipeline support this finding. The same related work (Huang et al., 2009) reports that systems viewed as a re-ranker work clearly outperforms classifier based approaches, suggesting a re-ranking was a better implementation choice.

(Bikel et al., 2009) designed inference rules to improve the performance of slot filling. We followed their idea but incorporated inference rules into Markov Logic Networks (MLN).

7 Conclusion

We developed three effective pipelines for the KBP slot filling task. We advance state-of-the-art performance using several novel approaches including statistical answer re-ranking and cross-slot reasoning based on Markov Logic Networks (MLN). Experimental results demonstrated that the pattern matching pipeline outperforms the top system in KBP2009 evaluation.

Acknowledgement

This work was supported by the U.S. Army Research Laboratory under Cooperative Agreement Number W911NF-09-2-0053, the U.S. NSF CAREER Award under Grant IIS-0953149, Google, Inc., DARPA GALE Program, CUNY Research Enhancement Program, PSC-CUNY Research Program, Faculty Publication Program and GRTI Program. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

Dan Bikel, Vittorio Castelli, Radu Florian and Ding-jung Han. 2009. Entity Linking and Slot Filling through Statistical Processing and Inference Rules. *Proc. TAC 2009 Workshop.*

Huang, Z., Thint, M., and Celikyilmaz, A. 2009. Investigation of question classifier in question answering. *Proc. ACL 2009*.

Heng Ji, Ralph Grishman, Zheng Chen and Prashant Gupta. 2009. Cross-document Event Extraction, Ranking and Tracking. *Proc. RANLP 2009*.

S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, and P. Domingos. 2007. The Alchemy system for statistical relational AI. *Technical report*, Department of Computer Science and Engineering, University of Washington.

Fangtao Li, Zhicheng Zheng, Fan Bu, Yang Tang, Xiaoyan Zhu and Minlie Huang. 2009. THU QUANTA at TAC 2009 KBP and RTE Track. *Proc. TAC 2009 Workshop.*

Bernardo Magnini, Matteo Negri, Roberto Prevete and Hristo Tanev. 2002. Mining Knowledge from Repeated Co-occurrences: DIOGENE at TREC-2002. *Proc. TREC-2002*.

Anselmo Peñas, Álvaro Rodrigo, Felisa Verdejo. 2007. Overview of the Answer Validation Exercise 2007. *Working Notes of CLEF 2007.*

Ravichandran, D., Hovy, E., and Och, F. J. 2003. Statistical QA -classifier vs. re-ranker: what's the difference?. *Proc. ACL 2003 Workshop on Multilingual Summarization and Question Answering*.

Matt Richardson and Pedro Domingos. 2006. Markov Logic Networks. *Machine Learning*. 62:107-136.

Nico Schlaefer, Jeongwoo Ko, Justin Betteridge, Guido Sautter, Manas Pathak, Eric Nyberg. 2007. Semantic Extensions of the Ephyra QA System for TREC 2007. *Proc. TREC 2007*.

Barry Schiffman, Kathleen McKeown, Ralph Grishman, and James Allan. 2007. Question Answering Using Integrated Information Retrieval and Information Extraction. *Proc. NAACL 2007*.